VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

DEPARTMENT OF CONTROL AND INSTRUMENTATION

HLASOVÝ GENERÁTOR PRO POKROČILÉ EMBEDDED SYSTÉMY

SPEECH GENERATOR FOR ADVANCED EMBEDDED SYSTEMS

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — Bc. Eliška Homzová

AUTHOR

VEDOUCÍ PRÁCE — Ing. Petr Petyovský, Ph.D.

SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Kybernetika, automatizace a měření**

Ústav automatizace a měřicí techniky

Studentka: Bc. Eliška Homzová ***ID:*** 211147

Ročník: 2 ***Akademický rok:*** 2022/23

NÁZEV TÉMATU:

Hlasový generátor pro pokročilé embedded systémy

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je navrhnout a realizovat software zajišťující generování hlasového výstupu.

1. Proveďte rešerši existujících principů generování hlasového výstupu v počítačových systémech. Definujte paměťové a výpočetní požadavky jednotlivých principů.
2. Zdokumentujte existující architektury generátorů hlasového výstupu pro embedded systémy.
3. Zvolte vhodný embedded systém pro realizaci knihovny hlasového výstupu. Definujte důvody pro jeho volbu i jeho případná omezení.
4. Navrhněte blokové schéma hlasového generátoru a implementujte jeho základní komponenty. Proveďte ověření funkčnosti jednotlivých bloků.
5. Dokončete programovou implementaci programu pro hlasový výstup v jazyce C/C++ a ověřte jeho funkčnost.
6. Upravte implementaci programu do formy SW knihovny pro zvolený embedded systém.
7. Navrhněte blokové uspořádání demonstrační aplikace tak, aby bylo možné hlasový generátor vhodně ovládat pomocí PC. Realizujte demonstrační aplikaci pro prezentaci funkčnosti výsledného řešení.
8. Zhodnoťte dosažené výsledky, uveďte výhody a nevýhody řešení. Navrhněte další možná rozšíření.

DOPORUČENÁ LITERATURA:

[1] JURAFSKY, Daniel a James H. MARTIN. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition [online]. 3rd ed. draft. 2023, 636 s. Dostupné z: <https://web.stanford.edu/~jurafsky/slp3/ed3book_jan72023.pdf>.

[2] ČADA, Ondřej. Mikroprocesor Motorola 68000: příručka programátora. Praha: Grada, 1992. ISBN 80-85424-64-9.

Termín zadání: 6.2.2023 ***Termín odevzdání:*** 17.5.2023

Vedoucí práce: Ing. Petr Petyovský, Ph.D.

doc. Ing. Petr Fiedler, Ph.D.
předseda rady studijního programu

ABSTRAKT

Cílem práce je navrhnout a realizovat software zajišťující generování hlasového výstupu, který bude tvořit softwarovou knihovnu pro zvolený embedded systém. Součástí práce je přehled existujících principů generování hlasového výstupu a analýza programu Atari 520ST Speech Synthesizer V2.0. Jako vhodný embedded systém byla vybrána vývojová deska Raspberry Pi Pico.

KLÍČOVÁ SLOVA

Atari 520ST, embedded systém, generátor řeči na základě textu, generátor řeči z textu, hlasový generátor, Raspberry Pi Pico, syntéza řeči

ABSTRACT

The aim of the master thesis is to design and implement a software for voice output generation, which will form a software library for the selected embedded system. The work includes a description of existing principles of voice output generation and an analysis of the Atari 520ST Speech Synthesizer V2.0 program. The Raspberry Pi Pico development board was selected as a suitable embedded system.

KEYWORDS

Atari 520ST, embedded system, Raspberry Pi Pico, speech synthesis, text to speech generator, voice generator

HOMZOVÁ, Eliška. *Hlasový generátor pro pokročilé embedded systémy*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky, 2023, 106 s. Diplomová práce. Vedoucí práce: Ing. Petr Petyovský, Ph.D.

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Bc. Eliška Homzová
VUT ID autora:	211147
Typ práce:	Diplomová práce
Akademický rok:	2022/23
Téma závěrečné práce:	Hlasový generátor pro pokročilé embedded systémy

Prohlašuji, že svou závěrečnou práci jsem vypracovala samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autorka uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušila autorská práva třetích osob, zejména jsem nezasáhla nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědoma následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno 17. 5. 2023 .....

podpis autorky*

*Autor podepisuje pouze v tištěné verzi.

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Cílem práce je navrhnout a realizovat software zajišťující generování hlasového výstupu, který bude tvořit softwarovou knihovnu pro zvolený embedded systém. Software zajišťující generování hlasového výstupu je nazýván hlasový generátor.

Hlasové generátory jsou programy nebo zařízení, jejichž úkolem je ze vstupních informací vytvořit syntetickou řeč, která je srozumitelná a zní přirozeně. Některé hlasové generátory generují syntetickou řeč, tak věrohodně, že umí napodobit hlas a způsob řeči skutečných osob [3] [4].

Úvodní část práce popisuje 3 základní metody generování hlasového výstupu v počítačových systémech a porovnává jejich výpočetní a paměťové požadavky. Dále popisuje syntézu řeči z textu, se kterou se v dnešní době setkáváme skoro na každém kroku, aniž bychom si to uvědomovali. Od počítačových her přes naše telefony až po zdravotní pomůcky [4].

Následující část práce se zabývá analýzou existující architektury generátoru hlasového výstupu, kterou je veřejně dostupný programu Atari 520ST Speech Synthesizer V2.0. Tento program převádí anglický text na syntetickou řeč [10].

Dále následuje popis vybraného embedded systému pro realizaci knihovny hlasového generátoru a důvodů pro jeho volbu.

V druhé polovině práce je navržen a implementován hlasový generátor pro vybraný embedded systém. Implementace hlasového generátoru je rozdělena na realizaci základních komponent, ze kterých je hlasový generátor složen a na dokončení jeho realizace.

Následně je ověřena funkcionalita implementovaného hlasového generátoru a vytvořena knihovna pro vybraný embedded systém.

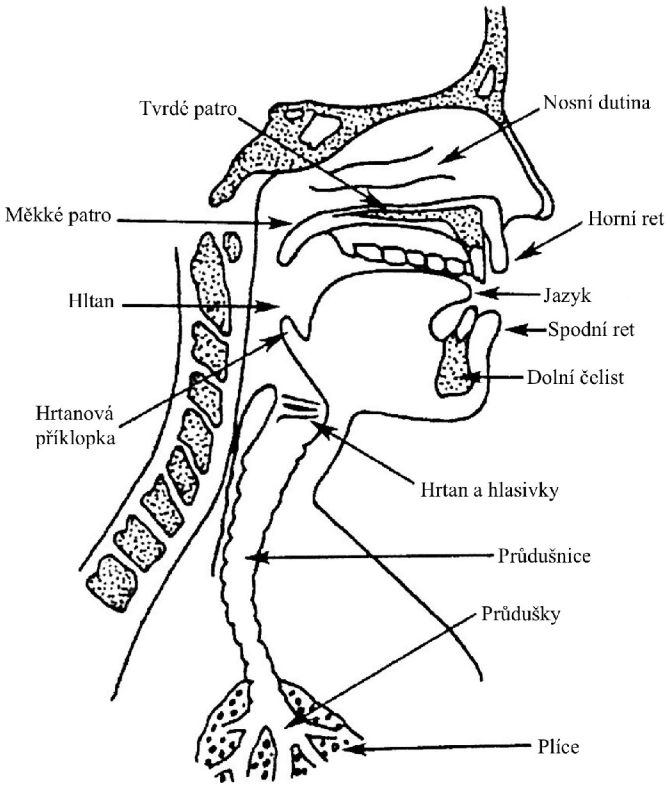
V poslední části práce je prezentována funkcionalita hlasového generátoru na vytvořených demonstračních aplikacích.

1 Generování hlasového výstupu v počítačových systémech

Následující text popisuje 3 základní metody generování hlasového výstupu a syntézu řeči z textu.

1.1 Úvod do tvorby mluvené řeči

Člověk vytváří většinu mluvené řeči vypuzováním vzduchu z plic přes průdušnici a hrtan s hlasivkami. Zvuky vytvářené hlasovým traktem se dělí na zvuky znělé, které jsou tvořeny chvěním hlasivek, a zvuky neznělé. Po průchodu hlasivkami vzduch opouští tělo ústy nebo nosní dutinou. Přitom většina zvuků je tvořena tím, že vzduch opouští tělo ústy [1]. Celý hlasový trakt je zobrazen na obrázku 1.1.



Obr. 1.1: Hlasový trakt člověka (upraveno z [5])

Výslovnost slov je popsána pomocí fonémů, které jsou definovány jako nejmenší lingvistické jednotky schopné rozlišit například slova, a které tvoří fonetické abecedy [1].

Příkladem fonetické abecedy jsou Mezinárodní fonetická abeceda IPA, která je mezinárodním standardem pro zápis výslovnosti světových jazyků, a jednoduchá fonetická abeceda ARPAbet, ve které jsou fonémy americké angličtiny tvořeny ASCII znaky [1]. V příloze A jsou vypsány symboly obou fonetických abeced pro anglické souhlásky a samohlásky.

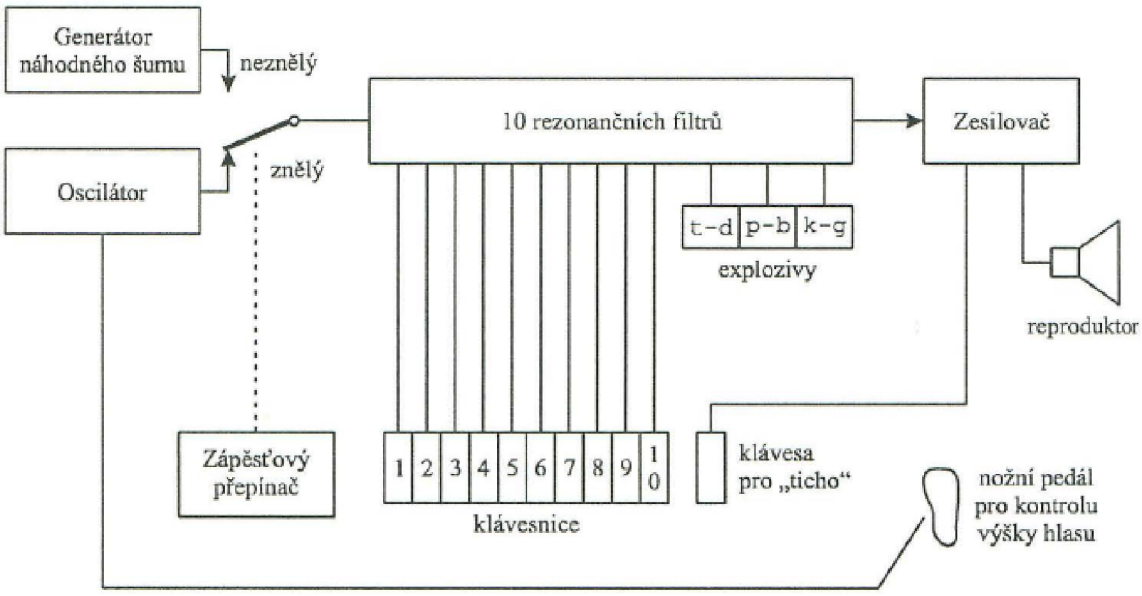
Fonémy se dělí do 2 hlavních skupin, kterými jsou souhlásky a samohlásky. Souhlásky vznikají omezením nebo zablokováním proudění vzduchu v hlasovém traktu a mohou být znělé i neznělé. Samohlásky jsou většinou znělé, více hlasité než souhlásky a mají delší dobu trvání oproti souhláskám [1].

Souhlásky se dále dělí do skupin podle místa a způsobu artikulace. Místo artikulace je místo v hlasovém traktu, kde dochází k omezení výdechového proudu. Způsob artikulace určuje jestli je foném tvořen úplným zablokováním proudění vzduchu hlasovým traktem a nebo jenom jeho omezením [1].

Samohlásky se dělí do skupin podle pohybu jazyka a tvaru rtů. U jazyku je důležité umístění jeho nejvyššího bodu, při vytváření samohlásek. Podle jeho výšky se určuje výška samohlásky a podle toho jestli je nejvyšší bod umístěn blíž ke rtům nebo k zadní části úst jsou samohlásky děleny na přední, zadní a střední. Samohlásky, při jejichž tvorbě dochází k velké změně umístění jazyku jsou nazývány dvojhlásky [1].

1.2 The Voder

The Voder, celým názvem The Voice operation demonstrator, je 1. elektronický syntetizér souvislé řeči. Byl vytvořen Homerem Dudleym a představen byl v roce 1939 na světovém veletrhu v New Yorku. Ovládán byl zkušeným operátorem, který za pomoci klávesnice a nožního pedálu, ovládal 10 paralelně zapojených pásmových propustí. Pro tvorbu znělých zvuků byl použit periodický signál, jehož perioda šla nastavit pomocí nožního pedálu, a pro tvorbu neznělých zvuků byl použit šum, který byl vytvářen pomocí generátoru náhodného šumu. Pro tvorbu českých souhlásek p, b, t, d, k a g byli přidány doplňující filtry. Doplňující filtry byli například v angličtině používány pro tvorbu souhlásek p, d, j a ch. Pro zesilování jednotlivých pásmových propustí bylo použito 10 tlačítek na klávesnici a znělost se řídila pomocí přepínače. The Voder byl sice schopen vytvářet souvislou řeč, ale vygenerované řeči bylo těžké rozumět [6] [7] [8]. Blokové schéma je možné vidět na obrázku 1.2.



Obr. 1.2: Blokové schéma The Voder [7]

1.3 Artikulační syntéza

Artikulační syntéza je nejobecnější metodou syntézy řeči, která modeluje celý proces tvorby lidské řeči. Zároveň je jedinou metodou, která se zabývá modelováním hlasového ústrojí člověka [7].

Artikulační model se skládá z modelu hlasivek, artikulátorů a jejich pohybů. Mezi artikulátory patří například čelisti, rty a jazyk. Syntetická řeč je generována pomocí matematické simulace vydechovaného vzduchu a artikulačních modelů. Artikulačními parametry jsou polohy artikulačních orgánů jako je například velikost a tvar retní štěrbiny [7].

Tato metoda je nejméně vyvinutou metodou ze zmíněných způsobů generování řeči, i když by v budoucnu mohla díky svojí komplexnosti vytvářet nejpřirozenější řeč [9].

1.4 Konkatenační syntéza

Konkatenační syntéza funguje na principu řetězení řečových jednotek, což jsou malé segmenty nahrávek lidské řeči. Řečové jednotky mohou mít různou velikost od malých jednotek jako je trifon až po celá slova [5].

Konkatenační syntéza spojující řečové jednotky o velikosti slov umožňuje tvorbu kvalitní syntetické řeči, ale její použití je omezené na aplikace s malým počtem slov [5] [9].

Mnohem více používané jsou konkatenační syntézy spojující řečové jednotky o velikosti difonu nebo demislabiky, z kterých lze složit jakékoli slovo. Problémem těchto řečových jednotek je to, že je nelze samostatně nahrát a proto je potřeba je získat z nahrávek souvislé řeči. Nahrávky souvislé řeči musí být nahrány 1 řečníkem, zřetelně a beze změny hlasu [5] [9].

Další věcí, která ovlivňuje přirozenost syntetické řeči je zachování přirozené koartikulace sousedních fonémů. Toho je dosaženo použitím například difonů, což jsou řečové jednotky, které začínají uprostřed 1. fonému a končí uprostřed následujícího. Nebo použitím demislabik, které začínají uprostřed 1. slabiky a končí uprostřed následující [5].

Řečové jednotky mohou být uloženy buď přímo jako nahrávky řeči a nebo jako parametry získané parametrizací nahrávky. Parametrizace má výhodu při minimalizování nesrovnalostí ve spojích řečových jednotek [9].

1.5 Formantová syntéza

Na rozdíl od konkatenační syntézy, při které dochází ke spojování řečových jednotek, je při formantové syntéze řeč tvořena pomocí akusticko-fonetických pravidel, které generují řídicí parametry syntezátoru [5].

Pro tvorbu akusticko-fonetických pravidel jsou použity nahrávky lidské řeči. Tato pravidla jsou často uložena v tabulkách, kde každá tabulka má v sobě uloženy hodnoty pro jeden foném nebo pro část fonému. Tabulka pro jeden foném nebo část fonému obsahuje, kromě formantových frekvencí, amplitud a dob trvání, i informace o tom jak vypočítat hodnoty frekvencí, amplitud a dob trvání mezi jednotlivými fonémy nebo částmi fonémů. Tyto hodnoty se můžou lišit podle toho jestli následuje souhláska nebo samohláska. Hodnoty pro samotný foném nebo část fonému se můžou lišit i podle toho, co mu předchází [5].

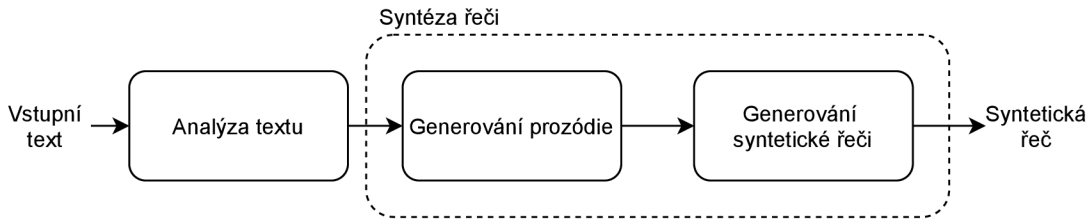
Problémem formantové syntézy je výběr vhodného počtu alofonů, které tvoří foném. Větší počet alofonů dělá syntetickou řeč přirozenější, takže je důležité nalézt jejich optimální počet [5].

Oproti konkatenační syntéze má formantová syntéza výhodu v tom, že může upravovat typ hlasu a rychlost řeči, aniž by museli být vytvořeny nové tabulky s akusticko-fonetickými pravidly. U konkatenační syntézy by museli být nahrány nové řečové jednotky [5].

1.6 Syntéza řeči z textu

Při syntéze řeči z textu je vstupem psaný text, který je zpracován a převeden na řeč za použití jedné z výše popsaných syntéz [5].

Syntéza řeči z textu se dá rozdělit do 2 částí a to na analýzu textu a na samotnou syntézu řeči. Syntéza řeči může být dále rozdělena na generování prozódie a generování syntetické řeči [5]. Na obrázku 1.3 je zobrazeno popsané blokové schéma syntézy řeči z textu.



Obr. 1.3: Blokové schéma syntézy řeči z textu (upraveno z [5])

V následujícím textu budou popsány jednotlivé části syntézy řeči z textu. Jako první bude popsána analýza textu, potom bude následovat generování prozódie. Generování syntetické řeči již bylo popsáno výše v rámci jednotlivých druhů syntézy.

1.6.1 Analýza textu

Analýza textu se skládá ze 7 částí, během kterých je provedeno předzpracování, analýza a převod textu na fonetický zápis.

Předzpracování

Prvním krokem v analýze textu je předzpracování, při kterém dochází k rozdělení textu do menších částí jako jsou například odstavce, věty nebo slova. Součástí předzpracování bývá i normalizace textu, která se využívá k nalezení speciálních znaků, zkratek, čísel, jednotek, malých a velkých písmen a k jejich následnému převedení do slovní formy [5].

Morfologická analýza

Po předzpracování textu následuje morfologická analýza. Při ní se slova rozdělí na jednotlivé morfémy. Mezi morfémy patří předpony, přípony, koncovky a kořeny slov. Například anglické slovo baseball obsahuje dva morfémy: "base"a "ball". Jedním z důvodů použití morfologické analýzy je zmenšení fonetického slovníku [5] [6].

Díky nalezení kořene slova nemusí slovník obsahovat všechny možné tvary daného slova. Takhle je ve fonetického slovníku uložen pouze kořen slova a na předpony, přípony, koncovky a další tvary slova jsou použita fonetická transkripční pravidla [5] [6].

Dalším důvodem použití morfologické analýzy může být v anglickém jazyce výskyt víceslovných slov a při předpovídaní neznámých slov. Morfologická analýza taky slouží pro získání informací, jako jsou například rod nebo slovní druh. Například anglické slovo "live" má různou výslovnost podle toho, o jaký slovní druh se jedná [5].

Fonetická transkripce

Výstup morfologické analýzy je použit ve fonetické transkripci, při které je text převeden do fonetického zápisu. Algoritmus tvorby fonetického zápisu začíná tak, že je nejprve vyhledáno ve fonetického slovníku celé slovo a pokud tam není, tak se hledají kořeny slov. Předpony, přípony, koncovky a časy jsou následně vytvořeny pomocí fonetických transkripčních pravidel. Fonetická transkripční pravidla se používají i pro tvorbu fonetického zápisu u neznámých slov [5].

Frekvence použití fonetického slovníku a fonetických transkripčních pravidel se může lišit podle jazyka, pro který je generátor tvořen. Například u angličtiny a dalších analytických jazyků, kde není moc slov vytvořených ze stejného kořene slova, se častěji používá fonetický slovník. Zatímco u češtiny a dalších flexivních jazyků, kde existuje hodně slov vytvořených ze stejného kořene, a ve kterých je velké množství slov, je vhodnější použít fonetické transkripční pravidla. V praxi se často používá kombinace obou způsobů převodu [7].

Syntakticko-prozodický rozbor

Cílem syntakticko-prozodického rozboru je vyřešení problému s více způsoby výslovnosti slova a rozdělení textu do frází. Problém s více způsoby výslovnosti je třeba v angličtině, kde má slovo jinou výslovnost podle toho zda je sloveso, podstatné nebo přídavné jméno. Tento problém nejde vyřešit bez analyzování skladby věty. Dále je potřeba poznat o jaký typ věty jde (oznamovací, rozkazovací, tázací), aby byla vytvořena správná prozódie [7].

Přiřazení lexikálního stresu

Dalším důležitým krokem při analýze textu je definování primárního a sekundárního stresu ve víceslabičných a víceslovných slovech. Pojmem lexikální stres je označován důraz na konkrétní slabiku ve slově nebo na konkrétní slovo ve frázi. Lexikální stres u slov vytvořených podle fonetických transkripčních pravidel lze zjistit podle dodatečných pravidel [5].

Lexikální stres může být přímo součástí fonetického slovníku a následně může být upraven podle dodatečných pravidel v případě rozšíření slova o předpony apod. Lexikální stres se může lišit i podle toho jakým slovním druhem slovo je [5].

1.6.2 Generování prozódie

Přirozenost syntetické řeči záleží na tom, jak dobře je vytvořena prozódie. Generování prozódie má 3 hlavní části, kterými jsou generování intenzity, časování a intonace. Intenzita je často spojena s druhem fonému a s umístěním slabiky ve slově. Intenzita má také nejmenší vliv na prozódii z vyjmenovaných hlavních částí [5] [6].

Generování časování

Jedním z důležitých kroků při generování prozódie je generování časování jednotlivých segmentů řeči. Při generování časování segmentu řeči dochází k nastavení délky trvání jednotlivých fonémů podle umístění ve slově a větě. Například slabiky na konci věty bývají delší než jinde ve větě. Důležitou roli hraje i v předchozím kroku definovaný primární a sekundární lexikální stres nebo rychlost řeči. Nastavení časování může být opět podle pravidel nebo může být součástí fonetického slovníku [5].

Kromě nastavování délky trvání fonémů dochází také k nastavení délky trvání pauz, které mohou být různě dlouhé. K určení jejich délky slouží informace získané při syntakticko-prozodickém rozboru [7].

Generování intonace

Dalším krokem při generování prozódie je generování intonace, při které dochází k tvorbě frekvence základního hlasivkového tónu. Správná intonace je důležitá v západních jazycích, jako je například angličtina nebo čeština, pro získání dodatečných informací jako jsou například druh věty nebo nálada mluvčího. Oproti tomu například v čínštině je intonace důležitá pro rozlišení významu podobných slabik [5] [7].

Obecně je generování intonace rozděleno do 2 částí. V 1. části je sestaven symbolický popis tvaru intonace a ve 2. části je následně z tohoto popisu vytvořena posloupnost hodnot frekvence základního hlasivkového tónu [5].

1.7 Srovnání základních metod generování hlasového výstupu

V této kapitole byli popsány 3 základní metody generování hlasového výstupu v počítačových systémech a syntéza řeči z textu při, které se využívají tyto základní metody.

Jako první byla popsána artikulační syntéza, která je nejobecnější metodou generování řeči. Artikulační modely jsou velmi složité a výpočetně náročné. Paměťová náročnost artikulační syntézy je mezi paměťovou náročností formantové a konkatenační syntézy [7] [9].

Následně byla popsána konkatenační syntéza, která tvoří syntetickou řeč spojováním řečových jednotek vytvořených z nahrávek lidské řeči. Vytvořená syntetická řeč je velice kvalitní a přirozená. Nevýhodou konkatenační syntézy může být vyšší výpočetní náročnost a velké nároky na velikost paměti, která může být podle zvolené řečové jednotky desítky až tisíce MB [5] [7].

Poslední popsanou základní metodou byla formantová syntéza, která generuje řeč pomocí akusticko-fonetických pravidel. Formantová syntéza taky vytváří kvalitní syntetickou řeč, ale oproti konkatenační syntéze zní méně přirozeně. Výhodou formantské syntézy je malá paměťová náročnost, která se pohybuje maximálně v desítkách kB [5] [7].

2 Architektura generátoru hlasového výstupu pro embedded systémy

Následují kapitola popisuje architekturu veřejně dostupného programu Atari 520ST Speech Synthesizer V2.0 z roku 1986 od autorů A. D. Beveridge a M. N. Day. Program Atari 520ST Speech Synthesizer V2.0 převádí psaný text na řeč a byl vytvořen pro osobní počítač Atari 520ST [10].

K tomuto programu již nejsou k dispozici zdrojové texty, takže bylo potřeba provést zpětnou analýzu strojového kódu převedeného do jazyka Assembler 68000. K převodu strojového kódu byl použit volně dostupný nástroj Ghidra [11]. Program byl dále analyzován pomocí emulátoru Hatari [12], který emuluje počítač Atari 520ST, a dostupné dokumentace k počítači Atari 520ST. Strojový kód programu je možné nalézt zde [10].

2.1 Atari 520ST

Atari 520ST je první 16bitový osobní počítač z řady osobních počítačů Atari ST od firmy Atari Corporation, který byl uveden na trh v roce 1985. Je to také první osobní počítač od firmy Atari Corporation, který je možné ovládat myší, a který má grafické uživatelské prostředí. Obsahuje mikroprocesor MC68000 a 512 kB RAM paměti. Atari 520ST má svůj vlastní operační systém Atari TOS, který je uložen v ROM paměti, což umožňuje rychlejší spuštění počítače a uvolnilo to místo v paměti RAM. Dále je počítač vybaven zvukovým obvodem YM2149 od firmy YAMAHA, který byl dříve vyráběn pod názvem AY-3-8910 firmou General Instrument. Kromě zvukového obvodu má počítač i 2 MIDI porty, díky nimž je i v dnešní době oblíbený v nahrávacích studiích. MIDI a sériové rozhraní řídí obvod MC6850 ACIA a paralelní rozhraní obsluhuje víceúčelová periferie MC68901 [13].

2.1.1 Mikroprocesor MC68000

MC68000 je nejstarší mikroprocesor architektury M68000 16/32bitových mikroprocesorů od firmy Motorola Inc. Obsahuje 16 32bitových datových (D0-D7) a adresových registrů (A0-A6). Dále obsahuje 32bitový čítač programu a 8bitový stavový registr. Datová sběrnice má 16 bitů a adresová sběrnice má 24 bitů. MC68000 má i svoji instrukční sadu [14].

2.1.2 Víceúčelová periferie MC68901

MC68901 je víceúčelová periferie od firmy Motorola Inc, která komunikuje přímo s mikroprocesorem MC68000 pomocí asynchronní sběrnice [15].

MC68901 obsahuje 8 samostatně programovatelných vstupně/výstupních pinů se schopností obsluhy přerušení, 4 8bitové časovače, z nichž 2 mají nastavitelný režim, a plně duplexní asynchronní sériovou komunikaci USART. Frekvence časovačů je 2,4576 MHz a je tvořena pomocí externího krystalového oscilátoru [16] [15].

2.1.3 Zvukový obvod AY-3-8910

Zvukový obvod AY-3-8910 je 3kanálový paměťově orienotovaný programovatelný generátor zvuku od firmy General Instrument, který se používal například v arkádových hrách, osobních počítačích nebo při tvorbě speciálních efektů. Kromě generování zvuku bylo možné zvukový obvod použít i pro připojení periferních zařízení. Podrobný popis zvukového obvodu AY-3-8910 je uveden v [17] [18].

2.2 Tabulky hodnot

Program obsahuje 7 tabulek, které jsou postupně popsány v rámci následujících podkapitol. Jediná tabulka, která nemá vlastní podkapitolu je tabulka umístění znaků. V této tabulce jsou uloženy hodnoty pro výpočet umístění začátku sekce jednotlivých znaků anglické abecedy v tabulce pangea.

2.2.1 Tabulka typtab

Tabulka typtab obsahuje 96 hodnot, které určují, do které kategorie patří daný ASCII znak. Hodnota pro daný ASCII znak je umístěna na pozici odpovídající hodnotě ASCII znaku. ASCII znaky jsou rozděleny do 11 skupin, které se liší v hodnotě, která je jim přiřazena. Rozdělení do skupin je popsáno v tabulce 2.1.

2.2.2 Tabulka special

Tato tabulka obsahuje informace pro převod speciálních znaků, čísel 0 až 9 a pořadí 1 až 10 na fonetický zápis. Informace pro každý speciální znak jsou rozděleny pomocí hodnoty 0. První informací je ASCII hodnota daného speciálního znaku. Potom následují buď ASCII znaky, které mohou ležet před nebo za daným speciálním znakem.

Tab. 2.1: Rozdělení ASCII znaků do 11 skupin v tabulce typtab

Hodnota	ASCII znaky
0_{16}	NUL SOH STX ETX EOT ENQ ACK BEL BS HT LF VT FF CR SO SI DLE DC1-DC4 NAK SYN ETB CAN EM SUB ESC FS GS RS US ()[\^_
40_{16}	Mezera ! "# $ % & ' * + , - . / : ; < = > ? @]
80_{16}	0 1 2 3 4 5 6 7 8 9
3_{16}	A E I O U Y
15_{16}	B M V W
D_{16}	C X
35_{16}	D N R
5_{16}	F H K P Q
$1D_{16}$	G
$3D_{16}$	J Z
$2D_{16}$	S
25_{16}	L T

Kromě hodnot konkrétních ASCII znaků, zde mohou být i hodnoty 0 až 9, které jsou při převodu textu na fonetický zápis používány pro označení, která kategorie ASCII znaků může ležet před nebo za daným speciálním znakem. Poslední informací u každého speciálního znaku je jeho fonetický zápis.

Ke každé hodnotě, kromě fonetického zápisu, může být přičtena hodnota 128, která je použita pro přechod mezi kontrolou předchozího a následujícího ASCII znaku v textu. V tabulce 2.2 je zobrazen příklad informací pro převod 1 speciálního znaku na fonetický zápis.

Tab. 2.2: Příklad informací pro převod jednoho speciálního znaku v tabulce special

S hodnotou 128	$A6_{16}$	80_{16}	80_{16}	20_{16}	41_{16}	45_{16}	$4E_{16}$	44_{16}	0_{16}
Bez hodnoty 128	'&'	0_{16}	0_{16}	' '	'A'	'E'	'N'	'D'	0_{16}

2.2.3 Tabulka pangea

Tato tabulka má stejnou strukturu jako tabulka special, jenom je používána pro převod znaků anglické abecedy na fonetický zápis.

Informace pro převod znaků anglické abecedy jsou uloženy v tabulce, buď jako celá slova u často se vyskytujících slov. A nebo ve formě částí slov a samotných znaků, což umožňuje vytvoření jakéhokoli anglického slova.

Části slov, celá slova a samostatné znaky budou souhrnně nazývány slova. V tabulce 2.3 je zobrazen příklad informací pro převod celého slova, části slova a samotného znaku na fonetický zápis.

Tab. 2.3: Příklad informací pro převod znaků v tabulce pangea

ASCII znak										
S hodnotou 128	$C1_{16}$	80_{16}	80_{16}	41_{16}	45_{16}	0_{16}				
Bez hodnoty 128	'A'	0_{16}	0_{16}	'A'	'E'	0_{16}				
Část slova										
S hodnotou 128	41_{16}	CC_{16}	81_{16}	82_{16}	55_{16}	$4C_{16}$	0_{16}			
Bez hodnoty 128	'A'	'L'	1_{16}	2_{16}	'U'	'L'	0_{16}			
Celé slovo										
S hodnotou 128	$4E_{16}$	$4F_{16}$	$D7_{16}$	81_{16}	81_{16}	$4E_{16}$	41_{16}	57_{16}	32_{16}	0_{16}
Bez hodnoty 128	'N'	'O'	'W'	1_{16}	1_{16}	'N'	'A'	'W'	'2'	0_{16}

2.2.4 Tabulka fonémů

Tabulka fonémů obsahuje hodnoty pro 60 fonémů a začátek hodnot pro 1 foném je označen zkratkou fonému. Každý foném má 16 hodnot, kromě fonémů ploziv (fonémy CH, J), které mají 32 hodnot, a fonémů afrikátů (fonémy P, T, K, B, D, G), které mají 48 hodnot. Hodnoty jsou použity pro výpočet řídicích parametrů zvuků.

Tab. 2.4: Seznam fonémů v tabulce fonémů

Dvojhlásky	EY AY OY AW OW UW
Samohlásky	AR WX YX AE IY ER AO UX UH AH AA OH AX IX IH EH
Souhlásky	DH ZH CH LX RX SH NX TH /H V Z J L R W Y Q P T K B D G M N F S
Speciální fonémy	Mezera - . , ? UL UM UN IL IM IN

2.2.5 Tabulka amplituda_offset

Tabulka amplituda_offset obsahuje 2048 hodnot o velikosti slova (2 bajtů). Hodnoty v tabulce jsou použity pro výpočet ukazatele na prvek tabulky amplitud, kterým má začít zápis do registrů zvukového obvodu.

2.2.6 Tabulka amplitud

Tabulka amplitud obsahuje adresy 3 registrů zvukového obvodu AY-3-8910, které jsou použity pro zápis amplitud kanálů A, B a C ($R8_{16}$, $R9_{16}$, RA_{16}).

Dále obsahuje požadované hodnoty amplitud kanálů A, B a C. Po adresách registrů a hodnotách amplitud následují 2 bajty 0, které značí konec 1 zápisu do registrů zvukového obvodu. Tabulka má celkem 2048 hodnot o velikosti 1 bajt, takže celkový počet variant amplitud pro zápis do zvukového obvodu je 256. V tabulce 2.5 je zobrazena konstrukce dat pro 1 zápis do registrů zvukového obvodu.

Tab. 2.5: Konstrukce dat pro 1 zápis do registrů zvukového obvody

$R8_{16}$	C_{16}	$R9_{16}$	B_{16}	RA_{16}	9_{16}	0_{16}	0_{16}

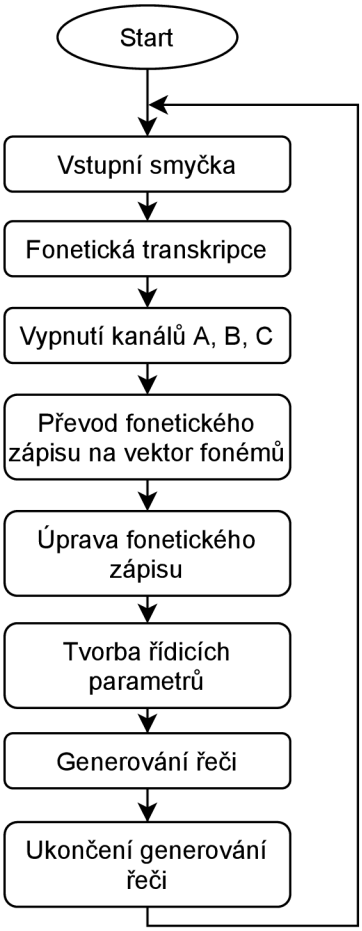
2.3 Popis programu

V následujícím textu je popsán algoritmus veřejně dostupného programu Atari 520ST Speech Synthesizer V2.0. Algoritmus programu je možné rozdělit do 8 částí, které jsou zobrazeny ve vývojovém diagramu na obrázku 2.1.

Algoritmus programu začíná vypsáním řetězce "Atari 520ST Speech Synthesizer V2.0\r\n"na obrazovku PC. Potom následuje vstupní smyčka programu, ve které je podle vstupního řetězce rozhodnuto, jestli následuje nastavení nové hodnoty tempa, intonace, režimu programu a nebo přechod na generování řeči.

2.3.1 Vstupní smyčka programu

Vstupní smyčka začíná přechodem na nový řádek na obrazovce PC. Následně je nastavena maximální velkost vstupního řetězce na 256 znaků a načten vstupní řetězce z obrazovky. Pokud je velikost vstupního řetězce rovna 0, tak je opět volána vstupní smyčka. Následně je na konec vstupního řetězce vložena 0.

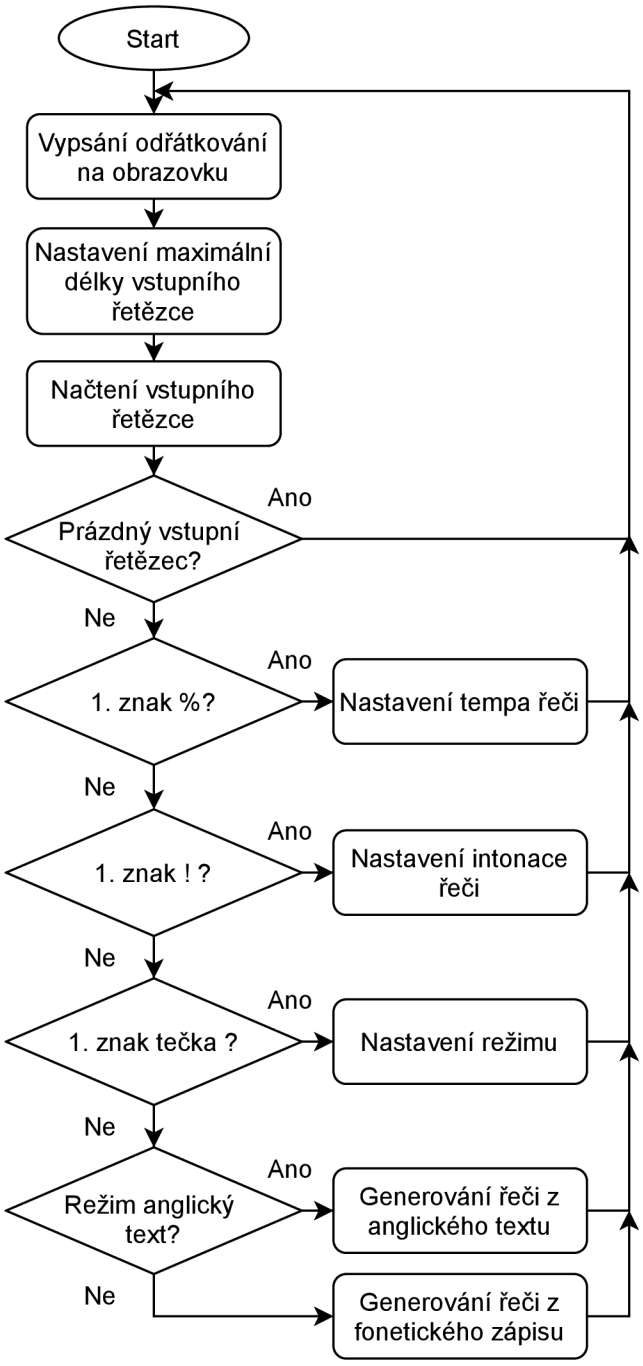


Obr. 2.1: Vývojový diagram algoritmu programu Atari 520ST Speech Synthesizer V2.0

Potom je ověřena hodnota 1. ASCII znaku vstupního řetězce. Jestli je 1. ASCII znak procentem, tak následuje nastavení tempa řeči. Pokud je 1. ASCII znak vykřičníkem, tak následuje nastavení intonace řeči. A když je 1. ASCII znakem tečka, tak je nastaven režim programu.

Když není 1. ASCII znak roven ani tečce, tak je otestováno nastavení režimu programu a podle toho, který režim je nastaven následuje fonetická transkripce nebo převod fonetického zápisu na vektor fonémů. Pokud je nastavený režim, při kterém je vstupem anglický text, tak je provedena fonetická transkripce. Jestli je nastaven režim, při kterém je vstupem fonetický zápis, tak je fonetická transkripce vynechána a následuje převod fonetického zápisu na vektor fonémů.

Vstupní smyčka je opět volána po dokončení generování řeči. Vývojový diagram vstupní smyčky programu je zobrazen na obrázku 2.2.



Obr. 2.2: Vývojový diagram vstupní smyčky programu

Nastavení tempa řeči

Nastavení tempa řeči začíná převodem ASCII znaků ve vstupním řetězci na číslo. Převod na číslo je proveden postupným načtením ASCII znaků, odečtením hodnoty 48 a jejich následným přičtením k proměnné mezivýsledek. Výchozí hodnotou proměnné mezivýsledek je 0 a před přičtením nové hodnoty je vynásobena číslem 10.

Pokud je hodnota nového ASCII znaku po odčítání záporná nebo větší než 9, tak je ukončen převod na číslo a proměnná mezivýsledek tvoří novou hodnotu tempa. Potom následuje ověření, zda je nová hodnota tempa v rozsahu 20 až 199. Výchozí hodnotou tempa řeči je 79.

Pokud nová hodnota tempa je v rozsahu 20 až 199, tak je nastavena nová hodnota tempa řeči a následně je volána vstupní smyčka programu. Když v něm není, tak program vygeneruje řeč pro chybové hlášení. Potom následuje návrat do vstupní smyčky programu. Hodnota 20 odpovídá rychlému tempu řeči a hodnota 199 pomalému tempu.

Nastavení intonace řeči

Nastavení intonace řeči probíhá stejným způsobem jako nastavení tempa řeči. Hodnota 48 odpovídá pronikavému zabarvení řeči, hodnota 80 normálnímu zabarvení řeči a hodnota 199 basovému zabarvení řeči. Výchozí hodnotou intonace řeči je 77.

Nastavení režimu programu

Program má 2 režimy. Prvním režimem je generování řeči ze vstupního anglického textu a 2. režimem je generování řeči z fonetického zápisu. První režim je nastaven pokud je proměnná mód kladná a 2. je nastaven pokud je proměnná mód záporná.

Mezi režimy se přechází pomocí bitové negace hodnoty proměnné mód. Po nastavení režimu je na obrazovku vypsána zpráva oznamující, který režim je momentálně nastaven.

2.3.2 Fonetická transkripce

Fonetická transkripce je provedena pokud je nastaven 1. režim programu. Začíná vyčištěním vstupního řetězce a potom následuje hlavní algoritmus převodu textu na fonetický zápis. Vývojový diagram fonetické transkripce je zobrazen na obrázku B.1.

Prvním krokem fonetické transkripce je čištění vstupního řetězce, při kterém jsou odstraněny nežádoucí znaky ve vstupním řetězci. Čištění řetězce začíná vložením mezery do vektoru vyčištěného textu a načtením 1. ASCII znaku vstupního řetězce. Hodnota ASCII znaku je potom porovnána s hodnotami 0 a 96. Pokud je hodnota ASCII znaku rovna 0, tak je 0 vložena do vektoru vyčištěného textu a je ukončeno čištění vstupního řetězce.

Když aktuální ASCII znak není roven 0 a je menší než 96, tak je nalezena odpovídající pozice v tabulce typtab a je ověřena její hodnota. V případě, že je hodnota rovna 0, následuje přechod na další ASCII znak vstupního řetězce. A když není rovna 0, tak je uložen aktuální ASCII znak do vektoru vyčištěného textu.

V případě, že je ASCII znak větší než 96, tak je proveden stejný postup jako by byl znak menší pouze je první proveden logický součin hodnoty aktuálního ASCII znaku s číslem 95, čímž dojde například k převodu malého písmene na písmeno velké.

Po vyčištění vstupního řetězce následuje převod textu na fonetický zápis, který je tvořen kontrolou ASCII znaků, které leží před a za aktuálním ASCII znakem.

Převod je prováděn postupným průchodem tabulkou pangea a kontrolou splnění informací jednotlivých slov. Následující postup je prováděn pro všechny ASCII znaky ve vektoru vyčištěného textu a nalezený fonetický zápis je uložen do vektoru fonetického zápisu.

Převod textu na fonetický zápis začíná načtením ASCII znaku z vektoru vyčištěného textu a následným otestováním bit 0 hodnoty, která je uložena v tabulce typetab na pozici odpovídající hodnotě aktuálního ASCII znaku. Pokud je bit 0 roven 0, tak je k aktuálnímu ASCII znaku přičtena hodnota 64. Jinak není přičteno nic a v obou případech následuje odečtení hodnoty 64 od aktuálního ASCII znaku.

Nakonec je vypočítaná hodnota vynásobena 2 a určuje umístění v tabulce umístění znaků. Hodnota z tabulky umístění znaků určuje začátek sekce aktuálního ASCII znaku v tabulce pangea. V případě speciálního znaku ukazuje umístění na začátek tabulky special.

Potom následuje tvorba fonetického zápisu podle aktuálního ASCII znaku a poloha aktuálního ASCII znaku ve vektoru vyčištěného textu je uložena do proměnné poloha, která je použita k pohybu ve vektoru vyčištěného text.

Dále je ověřeno, zda je aktuální ASCII znak roven 0. Pokud je roven 0, tak dojde k posunu na další slovo v tabulce pangea. Když ASCII znak není roven 0, tak je načten aktuální prvek slova v tabulce pangea. Bit 7 aktuálního prvku slova je následně nastaven do 0 a potom je porovnán s ASCII znakem, jehož umístění je dáno proměnnou poloha.

Pokud se nerovnají, tak opět následuje přechod na další slovo v tabulce pangea a proměnná poloha je zvětšena o 1. Když se rovnají, tak je ověřeno, jestli je neupravený aktuální prvek slova kladný a následuje načtení následujícího prvku aktuálního slova.

Když byl prvek slova kladný, tak je opakována předchozí část algoritmu pro nový ASCII znak, jehož umístění určuje proměnná poloha, a pro nový prvek aktuálního slova. Tímto způsobem je ověřeno, jestli za aktuálním ASCII znakem ve vektoru vyčištěného textu leží konkrétní ASCII znaky.

Pokud prvek slova nebyl kladný, tak je uložena poloha aktuálního ASCII znaku pro změnu do proměnné polohaPrefix a následuje tvorba fonetického zápisu podle prefixu, neboli tvorba fonetického zápisu podle ASCII znaků, které leží před aktuálním ASCII znakem.

Tvorba fonetického zápisu podle prefixu začíná načtením aktuálního prvku slova v tabulce pangea a je ukončena tehdy, když je aktuální prvek slova záporný nebo roven 0.

Po načtení aktuálního prvku slova následuje nastavení jeho bitu 7 do 0 a pokud výsledná hodnota není rovna 0, tak je porovnána s číslem 9. Když je menší, tak je volán podprogram kontrola prefixu, po jehož provedení může být přepínač Zero nastaven do 0 nebo do 1.

Pokud je nastaven do 1, tak prefix aktuálního ASCII znaku nesplňuje požadavek určený aktuálním prvkem slova v tabulce pangea a následuje návrat na začátek tvorby fonetického zápisu podle aktuálního ASCII znaku. A když je přepínač nastaven do 0, tak prefix aktuálního ASCII znaku splňuje požadavek určený aktuálním prvkem slova v tabulce pangea. V tomto případě následuje ověření zda má dojít k ukončení tvorby fonetického zápisu podle prefixu.

Pokud je upravená hodnota aktuálního prvku slova větší než 9, tak je proměnná polohaPrefix zmenšena o 1 a potom je porovnán ASCII znak ve vyčištěném vstupním řetězci, jehož poloha je určena proměnnou polohaPrefix, s upravenou hodnotou aktuálního prvku slova. Pokud se nerovnají, tak následuje přesun na následující slovo v tabulce pangea a návrat zpět na začátek tvorby fonetického zápisu podle aktuálního ASCII znaku.

Když se rovnají, tak následuje pouze kontrola ukončení tvorby fonetického zápisu podle prefixu. V případě, že je ukončena tvorba fonetického zápisu podle prefixu, tak následuje tvorba fonetického zápisu podle sufixu, neboli tvorba fonetického zápisu podle ASCII znaků, které leží za aktuálním ASCII znakem.

Před začátkem tvorby fonetického zápisu podle sufixu je uložena proměnná poloha do proměnné polohaSufix. Tvorba fonetického zápisu podle sufixu probíhá stejně jako tvorba fonetického zápisu podle prefixu, pouze je místo podprogramu kontrola prefixu volán podprogram kontrola sufixu a proměnná polohaSufix je na rozdíl od proměnné polohaPrefix zvětšována o 1.

Po tvorbě fonetického zápisu podle sufixu následuje uložení fonetického zápisu aktuálního slova v tabulce pangea do vektoru fonetického zápisu a proměnná poloha určuje, který ASCII znak ve vektoru vyčištěného textu bude novým aktuálním ASCII znakem.

Podprogram kontrola prefixu

V podprogram kontrola prefixu je ověřeno jestli ASCII znak, na který ukazuje proměnná polohaPrefix splňuje požadavky, které jsou určeny upravenou hodnotou aktuálního prvku slova v tabulce pangea. Upravená hodnota aktuálního prvku slova může být v rozsahu 1 až 8.

Jako první je snížena hodnota proměnné polohaPrefix o 1. Následně je v tabulce typetab nalezena hodnota, na kterou ukazuje ASCII znak ve vektoru vyčištěného textu určený proměnnou polohaPrefix.

Tato hodnota je uložena do proměnné typ a pomocí kontroly jednotlivých bitů její hodnoty jsou kontrolovány požadavky na ASCII znak. Informace o tom jaký bit proměnné typ a jaký požadavek je kontrolován jsou vypsány v tabulce 2.6.

V případě, že je požadavek splněn, tak je přepínač Zero nastaven do 0. Jinak je nastaven do 1. Pokud po ověření bitu 3 proměnné typ není splněn požadavek, že je ASCII znak jedním z ASCII znaků C, X, G, J, Z, S, tak je kontrolováno, zda je ASCII znakem H. V případě, že jím je a leží před ním ASCII znak C nebo S, tak je také splněn daný požadavek a příznak Zero je nastaven do 0.

Stejný postup probíhá při kontrole bitu 5 proměnné typ, pouze před ASCII znakem H můžou ležet ASCII znaky C, S nebo T.

Pokud je upravená hodnota prvku slova rovna číslu 7, tak nedochází ke kontrole bitů proměnné typ, ale přímo k porovnání ASCII znaku s ASCII znaky E, I, Y. A v případě, kdy je upravená hodnota prvku slova rovna číslu 8, dojde k zmenšování proměnné polohaPrefix o 1, dokud neukazuje na souhlásku ve vektoru vyčištěného textu. Když na ní ukazuje, tak je proměnná polohaPrefix zvýšena o 1 a vždy je nastaven přepínač Zero do 0.

Když je upravená hodnota prvku slova větší než 8, tak dojde k volaní chybové instrukce, která zahájí provádění výjimky číslo 4.

Tab. 2.6: Informace pro kontrolu požadavků v podprogramu kontrola prefixu

Upravená hodnota prvku slova	Ověřený bit	Požadavek na ASCII znak
1	0	Číslo, speciální znak
2	1	Samohláska
3	2	Souhláska
4	4	B, M, V, W, D, N, R, G, J, Z
5	3	C, X, G, J, Z, S
6	5	D, N, R, J, Z, C, S, L, T
7	-	E, I, Y
8	2	Není samohláska

Podprogram kontrola sufixu

V podprogram kontrola sufixu je ověřeno jestli ASCII znak, na který ukazuje proměnná polohaSufix splňuje požadavky, které jsou určeny upravenou hodnotou aktuálního prvku slova v tabulce pangea. Upravená hodnota aktuálního prvku slova může být v rozsahu 1 až 9.

Podprogram kontrola sufixu probíhá stejně jako podprogram kontrola prefixu, pouze proměnná polohaSufix není zmenšována, ale zvětšována o 1.

Dalším rozdílem je, že proměnná polohaSufix je zvětšena o 1 až potom co je vytvořená proměnná typ a upravená hodnota aktuálního prvku slova může být v rozsahu 1 až 9. Informace o tom jaký bit proměnné typ a jaký požadavek se kontroluje jsou vypsány v tabulce 2.7.

V případě, že je upravená hodnota aktuálního prvku slova rovna číslu 9, tak nedojde ke kontrole bitu proměnné typ, ale přímo ke kontrole ASCII znaků co následují za ASCII znakem určeným proměnnou polohaSufix.

Tab. 2.7: Informace pro kontrolu požadavků v podprogramu kontrola sufix

Upravená hodnota prvku slova	Ověřený bit	Požadavek na ASCII znak
1	0	Číslo, speciální znak
2	1	Samohláska
3	2	Souhláska
4	4	B, M, V, W, D, N, R, G, J, Z
5	3	C, X, G, J, Z, S
6	5	D, N, R, J, Z, C, S, L, T
7	-	E, I, Y
8	2	Není samohláska
9	-	-ING, -ER,-ES, -ED, -ELY, -EFUL

2.3.3 Vypnutí výstupních kanálů zvukového obvodu AY-3-8910

Před převodem fonetického zápisu na vektor fonémů jsou vypnuty kanály zvukového obvodu AY-3-8910. Jejich vypnutí je provedeno zápisem hodnoty $7F_{16}$ do registru $R7_8$ zvukového obvodu.

2.3.4 Převod fonetického zápisu na vektor fonémů

Vektor fonetického zápisu obsahuje seznam fonémů, které mohou být doplněny o dodatečné informace týkající se délky a tónu jednotlivých fonémů.

Každý foném může být doplněn o číslo v rozsahu 1 až 9, které upravuje tón fonému. Dále může být doplněn o znaménka < a >, které značí zda má být foném zkrácen nebo prodloužen. Parametry zkrácení a prodloužení jsou souhrnně nazývány parametry délky. V případě, že je nastaven 2. režim programu, tak vektor fonetického zápisu tvoří vstupní řetězec programu.

Při úpravě fonetického zápisu je postupně procházen vektor fonetického zápisu a zároveň je vytvářen vektor fonémů. Ten obsahuje hodnoty umístění fonému v tabulce fonémů, informace o tónu a délce fonému a počet zvuků, které tvoří daný foném. Informace o každém fonému jsou uloženy do 4 bajtů, jak je zobrazeno v tabulce 2.8. Na obrázku B.2 je zobrazen vývojový diagram převodu fonetického zápisu na vektor fonémů.

Tab. 2.8: Konstrukce informací jednoho fonému ve vektoru fonémů

Pořadí		Funkce
1. - 2. bajt		umístění fonému v tabulce fonémů
3. bajt	1. - 4. bit	tón
	7. bit	zkrácení
	8. bit	prodloužení
4. bajt		počet zvuků

Při převodu fonetického zápisu na vektor fonémů je každý ASCII znak nejdříve porovnán s hodnotami 49 a 58. Pokud je z daného rozsahu, tak je převeden na číslo a uložen do parametru tónu aktuálního fonému vektoru fonémů. Když ASCII znak není z rozsahu hodnot 49 a 58, tak je následně porovnán s hodnotou 62. Jestliže se jí rovná, tak je nastaven bit pro prodloužení do 1. Jinak se ASCII znak porovnává s hodnotou 60 a když se rovnají, tak je nastaven bit pro zkrácení do 1.

Jestli se nerovnal ani hodnotě 60 dojde k porovnání s hodnotami 97 a 123. Pokud je z daného rozsahu, tak je ASCII znak malé písmeno a je převeden na velké písmeno odečtením hodnoty 32. Když z rozsahu není, tak není odčítáno nic a ASCII znak je rovnou porovnáván s hodnotami v tabulce fonémů. Pokud je nalezena hodnota v tabulce fonémů, která je rovna ASCII znaku ve vektoru fonetického zápisu, tak je následující hodnota v tabulce fonémů porovnána nejprve s hodnotou 32, aby se zjistilo, jestli daný foném není tvořen jenom jedním ASCII znakem.

Když je daný foném tvořen jedním ASCII znakem, tak je ukončeno hledání fonému a pokud není tvořen jenom jedním ASCII znakem, tak je následující hodnota v tabulce fonémů porovnána s následujícím ASCII znakem ve vektoru fonetického zápisu. Jestli se i tyto dvě hodnoty rovnají, tak byl nalezen začátek hodnot pro výpočet parametrů daného fonému.

Nalezený začátek hodnot pro výpočet parametrů daného fonému je následně uložen do 1. a 2. bajtu aktuálního fonému vektoru fonémů.

Po ukončení vytváření vektoru fonémů je na jeho konec vložen foném Q a za něj hodnota 0. Za 0 je vložena dvakrát hodnota $FFFF_{16}$. Pokud dojde k chybě během tvorby vektoru fonémů, tak následuje návrat do vstupní smyčky programu.

2.3.5 Úprava fonetického zápisu

Úprava fonetického zápisu je rozdělena do 5 částí, které upravují nebo přidávají fonémy do vektoru fonémů podle umístění fonémů ve slově.

Změna fonémů podle umístění ve slově

Změna fonémů podle umístění ve slově mění foném podle toho, které fonémy leží před ním a za ním. Dále přidává za dvojhlásky jejich ukončovací foném.

Postup změny fonémů začíná ověřením, zda je aktuální foném vektoru fonémů mezera. Pokud je mezerou, tak následuje přechod na další foném vektoru fonémů. Jestliže je aktuální foném jedním z fonémů UL, UM, UN, IL, IM, IN, tak je hodnota umístění v tabulce fonémů změněna na hodnotu, která je uložena v 1. hodnotě aktuálního fonému v tabulce fonémů. Dále je za aktuální foném vložen další foném jehož hodnota umístění je rovna 2. hodnotě aktuálního fonému v tabulce fonémů a vložený foném má stejný tón a délku jako aktuální foném.

V případě, že aktuálním fonémem je jeden z fonémů UL, UM, UN, tak 1. hodnota ukazuje na foném AX a 2. hodnoty ukazují na fonémy L, M, N. Když je aktuální foném jedním z fonémů IL, IM, IN, tak 1. hodnota ukazuje na foném IX a 2. hodnoty ukazují na fonémy L, M, N.

Pokud je aktuální foném jednou z dvojhlásek EY, AY, OY, tak je za něj vložen foném YX se stejným tónem a délkou jako má dvojhláska. A když je aktuální foném jednou z dvojhlásek AW, OW, UW, tak je za něj vložen foném WX.

Jestliže je aktuální foném fonémem R a předchozí foném je samohláskou nebo dvojhláskou a následující foném není samohláska ani dvojhláska, tak je foném R nahrazen fonémem RX.

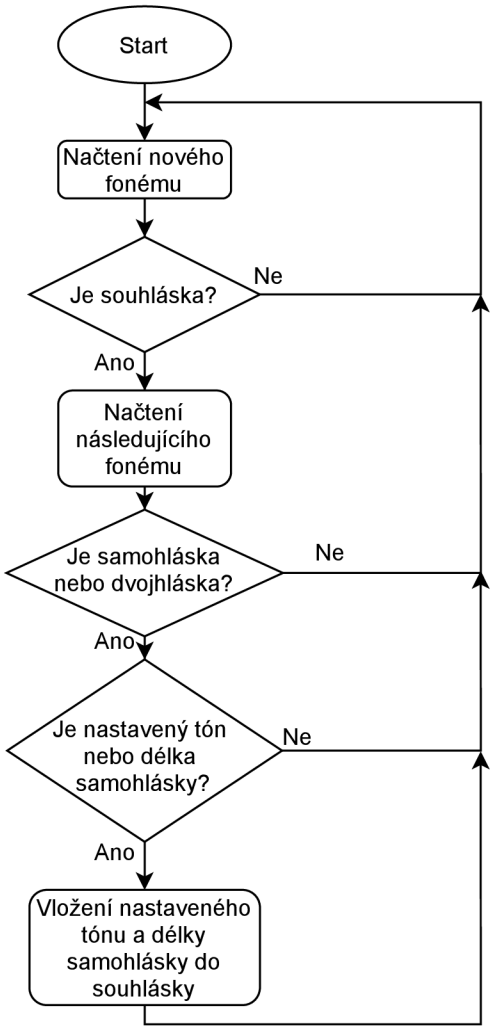
Pokud aktuální foném není fonémem R, tak je ověřeno jestli není fonémem L. Když je fonémem L a předchozí foném byla samohláska nebo dvojhláska a následující foném není samohláska ani dvojhláska, tak je nahrazen fonémem LX.

Jestliže není fonémem L, tak je ověřeno zda není fonémem S. Pokud jím je, tak je zjištěno, zda je předchozí foném fonémem G. Pokud předchozí foném je foném G, tak je foném S nahrazen fonémem Z. Jinak je ověřeno, jestli není předchozím fonémem foném P, který je změněn na foném B, pokud předchozím fonémem byla samohláska nebo dvojhláska.

Pokud předchozím foném není fonémem P, ale fonémem T a je předním samohláska nebo dvojhláska, tak je nahrazen fonémem D. Když předchozí foném není T, ale foném K, před kterým je samohláska nebo souhláska, tak je změněn na foném G. Pokud není ani fonémem K, tak není provedena žádná změna a následuje kontrola dalších fonémů vektoru fonémů. Vývojový diagram změny fonémů podle umístění ve slově je zobrazen na obrázku B.3.

Úprava tónu a délky souhlásky

Při úpravě tónu a délky souhlásky je změněn tón a délka souhlásky podle následující samohlásky nebo dvojhlásky. Na obrázku 2.3 je zobrazen vývojový diagram popisující postup úpravy 1 souhlásky.



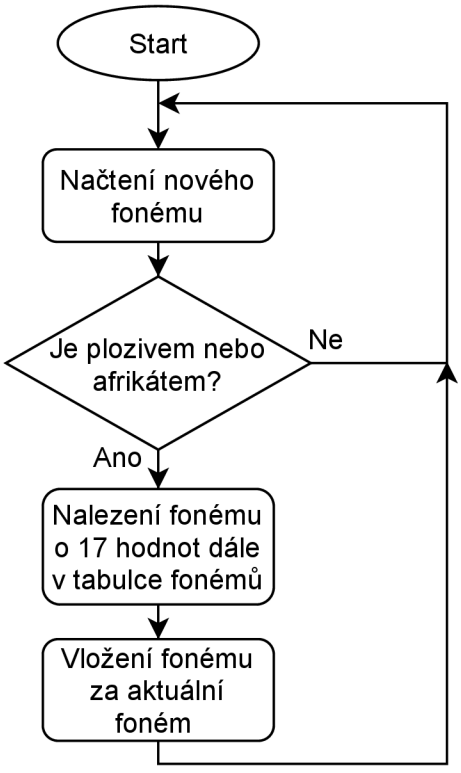
Obr. 2.3: Vývojový diagram úpravy tónu a délky souhlásky

Úprava tónu a délky souhlásky začíná nalezením souhlásky a ověřením následujícího fonému vektoru fonémů. Pokud je následující foném samohláska nebo dvojhláska, tak je zkontrolováno, zda má nastavený tón nebo délku. Když má nastaven alespoň jeden z těchto parametrů, tak jsou tyto parametry uloženy do odpovídajících parametrů souhlásky.

Úprava ploziv a afrikátů

Úprava ploziv a afrikátů začíná kontrolou bitu 7 2. bajtu 16. hodnoty aktuálního fonému v tabulce fonémů. Pokud je roven 1, tak je foném plozivem nebo afrikátem. A když je roven 0, tak následuje kontrola dalšího fonému vektoru fonémů.

Když je foném plozivem nebo afrikátem, tak je za aktuální foném vložen foném nový s hodnotou umístění v tabulce fonémů posunutou o 17 hodnot tabulky fonémů dopředu a se stejnou hodnotou tónu a délky. Vývojový diagram pro úpravu ploziv a afrikátů je zobrazen na obrázku 2.4.



Obr. 2.4: Vývojový diagram úpravy ploziv a afrikátů

Úprava fonémů podle nastaveného tónu a délky

Při úpravě fonémů podle nastaveného tónu a délky jsou vypočítány počty zvuků, ze kterých jsou složeny jednotlivé fonémy.

Úprava fonémů podle nastaveného tónu a délky začíná ověřením, zda má aktuální foném vektoru fonémů nastavené prodloužení. Pokud ho má nastavené, tak je 1. bajt 1. hodnoty aktuálního fonému v tabulce fonémů logicky posunut o 1 bit doprava a je k němu přičteno číslo 1. Následně je přičtena i jeho původní hodnota. Takto vypočítaná hodnota je uložena do počtu zvuků aktuálního fonému.

Když nemá nastavené prodloužení, tak následuje ověření nastavení zkrácení. Jestliže je nastaveno, tak dojde k logickému posunu 1. bajtu 1. hodnoty aktuálního fonému v tabulce fonémů o 1 bit doprava a k přičtení 1. Výsledek je následně uložen do počtu zvuků aktuálního fonému.

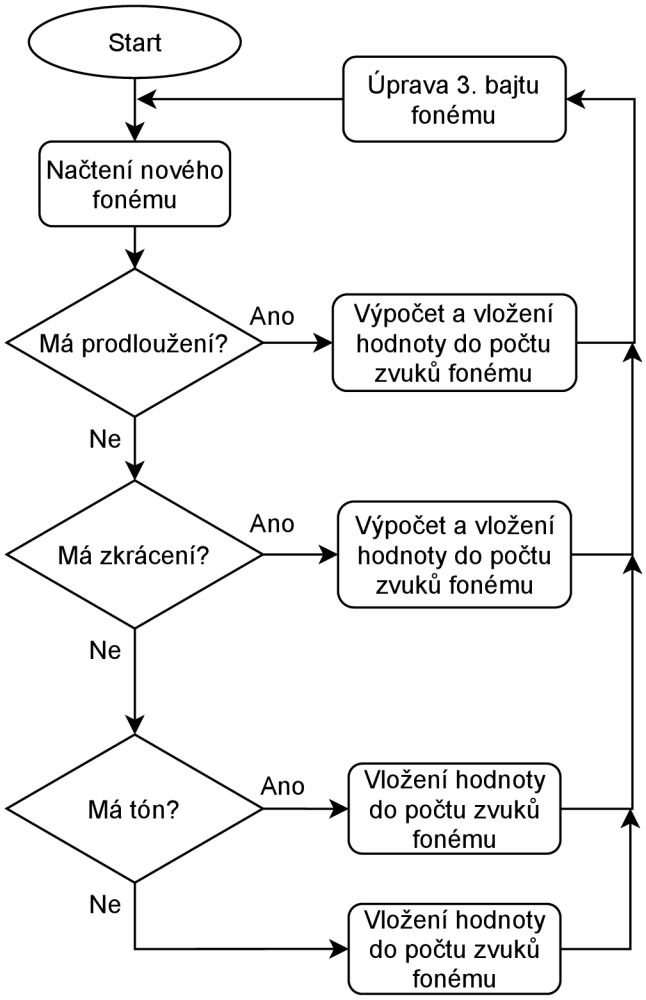
Pokud není nastaveno ani zkrácení, ale je nastaven tón fonému, tak je 2. bajt 1. hodnoty aktuálního fonému přímo uložen do počtu zvuků aktuálního fonému. Když není nastaven ani tón fonému, tak je do počtu zvuků aktuálního fonému uložen 1. bajt 1. hodnoty aktuálního fonému.

Ať už má aktuální foném jakékoli nastavení tónu a délky, vždy je k parametrům tónu a délky bitově přičtena hodnota 63. Vývojový diagram je zobrazen na obrázku 2.5.

Úprava souhlásky před speciálním fonémem

Při úpravě souhlásky před speciálním fonémem je nejprve zjištěno, jestli je aktuální foném vektoru fonémů speciální foném. Pokud není speciálním fonémem, tak následuje přechod na další foném vektoru fonémů.

Když jím je, tak je ověřeno, jestli je předchozí foném souhláska. Pokud jí je, tak je zkontrolován další předchozí foném. Takhle se postupuje vektorem fonémů nazpět, dokud není nalezen foném, který není souhláskou. Až je nalezen, tak dojde k návratu o jeden foném napřed a k upravení hodnoty jeho počtu zvuků. Hodnota je upravena tak, že jsou logicky posunuty bity hodnoty počtu zvuků o 1 doprava a následně je k němu přičtena 1. Stejně jsou upraveny všechny fonémy až po aktuální foném vektoru fonémů. Úprava není provedena pro souhlásky S, F, TH, SH, CH, což jsou souhlásky, které mají bit 5 2. bajtu 16. hodnoty roven 1 a bit 6 2. bajtu 16. hodnoty roven 0. Na obrázku B.4 je zobrazen vývojový diagram úpravy souhlásky před speciálním fonémem.



Obr. 2.5: Vývojový diagram úpravy fonémů podle nastaveného tónu a délky

2.3.6 Tvorba řídicích parametrů

Tvorba řídicích parametrů je rozdělena do 6 částí, které postupně vytváří vektor řídicích parametrů, ve kterém je každých 9 parametrů použito pro zápis jednoho zvuku generované řeči. V tabulce 2.9 jsou popsány funkce jednotlivých parametrů pro zápis 1 znělého nebo neznělého zvuku. Řídicí parametry jsou tvořeny postupným zpracováním hodnot fonémů uložených ve vektoru fonémů. Na obrázku B.5 je zobrazen vývojový diagram tvorby řídicích parametrů.

Součástí tvorby řídicích parametrů je vektor proměnných, do kterého jsou ukládány vybrané hodnoty fonémů z tabulky fonémů. Vektor proměnných tvoří 16 slov (32 bajtů) a poslední dvě slova mají hodnotu $ffff_{16}$.

Tab. 2.9: Konstrukce řídicích parametrů pro 1 zvuk generované řeči

Parametr	Funkce
1.	Změna výpočtu umístění v tabulce amplituda_offset u znělého zvuku
2.	Amplituda kanálu C u neznělého zvuku
3.	Perioda generátoru šumu u neznělého zvuku
4.-9.	Parametry pro výpočet umístění v tabulce amplituda_offset

Tvorba řídicích parametrů začíná vynulováním vektoru řídicích parametrů. Následně je vynulována 11. pozice vektoru proměnných a na vybrané pozice vektoru proměnných jsou vloženy vybrané hodnoty aktuálního fonému z tabulky fonémů. Podrobný popis vybraných hodnot aktuálního fonému a jejich pozice ve vektoru proměnných je v tabulce 2.10.

Uložením vybraných hodnot aktuálního fonému do vektoru proměnných začíná cyklus výpočtu a úpravy 2., 4., 5., 6., 7., 8. a 9. řídicího parametru jednotlivých fonémů ve vektoru fonémů.

Tab. 2.10: Tabulka vybraných hodnot a jejich umístění ve vektoru proměnných na začátku tvorby řídicích parametrů

Umístění	Vložená hodnota
2. pozice	2. hodnota aktuálního fonému
5. pozice	4. hodnota aktuálního fonému
8. pozice	6. hodnota aktuálního fonému
1. bajt 3. pozice	1. bajt 10. hodnoty aktuálního fonému
1. bajt 6. pozice	1. bajt 11. hodnoty aktuálního fonému
1. bajt 9. pozice	1. bajt 12. hodnoty aktuálního fonému
1. bajt 10. pozice	1. bajt 13. hodnoty aktuálního fonému
13. pozice	vynulována
14. pozice	vynulována

Po vložení vybraných hodnot aktuálního fonému do vektoru proměnných je porovnán 2. bajt 11. pozice vektoru proměnných s počtem zvuků aktuálního fonému a pokud je počet zvuků aktuálního fonému větší, tak je do 2. bajtu 11. pozice vektoru proměnných vložena hodnota počtu zvuků. Stejný postup se opakuje pro hodnotu uloženou v 1. bajtu 11. pozice vektoru proměnných a následuje po něm výpočet 4., 6. a 8. parametru zvuku vektoru řídicích parametrů.

Výpočet 4., 6. a 8. parametru zvuku vektoru řídicích parametrů

Výpočet 4., 6. a 8. parametru zvuku je tvořen 1 algoritmem, který má pro všechny 3 parametry stejný průběh a liší se pouze v hodnotách vstupních proměnných.

Na začátku je nalezeno umístění ve vektoru řídicích parametrů, které značí začátek zápisu hodnot pro aktuální foném. Hodnota umístění je získána vynásobením 2. bajtu 13. pozice vektoru proměnných číslem 9. Hodnota umístění je pro všechny parametry stejná.

Algoritmus výpočtu 4., 6. a 8. parametru bude popsán na výpočtu 4. parametru zvuku. Algoritmus má 4 vstupní proměnné. Do 1. proměnné je vložena hodnota na 1. pozici vektoru proměnných a do 2. proměnné je vložena hodnota na 2. pozici. Do 3. proměnné je vložen 2. bajt 11. pozice vektoru proměnných a ve 4. proměnné je uloženo umístění 4. parametru aktuálního zvuku ve vektoru řídicích parametrů.

Jako první je ověřeno, jestli není 3. proměnná rovna 0. Pokud je rovna 0, tak není nenastaven 4. parametr zvuku. Jinak je od 2. proměnné odečtena 1. proměnná a nejnižší bajt výsledné hodnoty je rozšířen na dlouhé slovo. Následně je výsledná hodnota znaménkově vydělena 3. proměnnou. Výsledek po dělení je vložen do proměnné mezivýsledek dělení a nakonec je výsledná hodnota aritmeticky posunuta o 1 bit doprava a je k ní přičtena 1. proměnná. Výsledná hodnota je uložena do proměnné mezivýsledek.

Po výpočtu proměnné mezivýsledek následuje cyklus jehož počet opakování je dán hodnotou 3. proměnné, která je zmenšena o 1. Cyklus začíná vynásobení proměnné mezivýsledek číslem 16 a jejím logickým posunutím o 5 bitů doprava. Potom je ověřena hodnota 4. parametru zvuku.

Když je rovna 0, tak je do 4. parametru zvuku vložen výsledek logického posunu a k původní hodnotě proměnné mezivýsledek je přičtena hodnota proměnné mezivýsledek dělení a výsledná hodnota je vložena do proměnné mezivýsledek. Pokud hodnota 4. parametru není rovna 0, tak je otestována proměnná mezivýsledek dělení. Když proměnná mezivýsledek dělení není záporná, tak je porovnána hodnota 4. parametru s výsledek logického posunu. Pokud je výsledek logického posunu větší, tak je hodnota výsledku logického posunu vložena do 4. parametru a následně je vypočítána nová hodnota proměnné mezivýsledek. Když není větší, tak je pouze vypočítána nová hodnota proměnné mezivýsledek.

Jestliže proměnná mezivýsledek dělení je záporná a hodnota výsledku logického posunu je větší nebo rovna hodnotě 4. parametru, tak je vypočítána pouze nová hodnota proměnné mezivýsledek. Jinak je vložena hodnota výsledku logického posunu do 4. parametru a vypočítána nová hodnota proměnné mezivýsledek.

Nakonec je 3. proměnná snížena o jedna a ke 4. proměnné je přičteno číslo 9, čímž dojde k přesunu na další zvuk vektoru řídicích parametrů. Cyklus je opakován dokud není 3. proměnná záporná.

Popsaný algoritmus se opakuje pro výpočet 6. a 8. parametru s tím, že se liší hodnoty vstupních proměnných, které jsou vypsány v tabulce 2.11. Vývojový diagram výpočtu 4. parametru zvuku je zobrazen na obrázku B.6.

Tab. 2.11: Tabulka hodnot vstupních proměnných pro výpočet 4., 6. a 8. parametru zvuku

	4. parametr	6. parametr	8. parametr
1. proměnná	1. pozice	4. pozice	7. pozice
2. proměnná	2. pozice	5. pozice	8. pozice
3. proměnná	2. bajt 11. pozice	2. bajt 11. pozice	2. bajt 11. pozice
4. proměnná	Umístění 4. parametru	Umístění 6. parametru	Umístění 8.parametru

Výpočet 2., 5., 7. a 9. parametru zvuku vektoru řídicích parametrů

Výpočet 2., 5., 7. a 9. parametru zvuku vektoru řídicích parametrů je stejně jako výpočet 4., 6. a 8. parametru zvuku tvořen 1 algoritmem, který má pro všechny 4 parametry stejný průběh a liší se pouze v hodnotách vstupních proměnných. Nejprve je nalezeno umístění ve vektoru řídicích parametrů, které značí začátek zápisu hodnot pro aktuální foném. Hodnota umístění je získána vynásobením 2. bajtu 14. pozice vektoru proměnných číslem 9. Hodnota umístění je pro všechny parametry stejná.

Algoritmus je stejný jako algoritmus pro výpočet 1 parametru ve výpočtu 4., 6. a 8. parametru zvuku vektoru řídicích parametrů, pouze se liší počáteční postup výpočtu proměnných mezivýsledek dělení a mezivýsledek. Dále se liší úprava proměnné mezivýsledek na konci jednoho běhu cyklu a výpočet hodnoty, která je vkládána do zvoleného parametru.

Výpočet počáteční hodnoty proměnné mezivýsledek začíná logickým součinem 1. proměnné s hodnotou 255. Výsledek je následně odečten od hodnoty 2. proměnné a výsledná hodnota je logicky posunuta o 8 bitů doleva. Po posunu o 8 bitů doleva je výsledná hodnota rozšířena na dlouhé slovo a vydělena hodnotou 3. parametru. Po dělení opět následuje rozšíření na dlouhé slovo a logické posunutí o 8 bitů doleva. Výsledek je uložen do proměnné mezivýsledek dělení.

Hodnota proměnné mezivýsledek dělení je potom aritmeticky posunuta o 1 bit doprava a je k ní přičtena hodnota $ff0000_{16}$. Nakonec je prohozeno 1. a 2. slovo výsledné hodnoty a hodnota je vložena do proměnné mezivýsledek.

Výpočet hodnoty, která je vkládána do zvoleného parametru začíná vynásobením 1. bajtu proměnné mezivýsledek číslem 3 a následným odečtením čísla 89 od výsledku násobení. Pokud není výsledek odčítání kladný, tak je nastaven na 0. Výsledek odčítání je nakonec logicky posunut o 1 doprava.

Úprava proměnné mezivýsledek na konci jednoho běhu cyklu začíná prohozením 1. a 2. slova hodnoty proměnné mezivýsledek. Následně je k hodnotě přičtena hodnota proměnné mezivýsledek dělení a opět následuje prohození 1. a 2. slova výsledné hodnoty. Hodnoty vstupních proměnných pro výpočet jednotlivých parametrů jsou vypsány v tabulce 2.12.

Tab. 2.12: Tabulka hodnot vstupních proměnných pro výpočet 2., 5., 7. a 9. parametru zvuku

	2. parametr	5. parametr	7. parametr	9. parametr
1. proměnná	2. bajt 10. pozice	2. bajt 3. pozice	2. bajt 6. pozice	2. bajt 9. pozice
2. proměnná	1. bajt 10. pozice	1. bajt 3. pozice	1. bajt 6. pozice	1. bajt 9. pozice
3. proměnná	1. bajt 11. pozice	1. bajt 11. pozice	1. bajt 11. pozice	1. bajt 11. pozice
4. proměnná	Umístění 2. parametru	Umístění 5. parametru	Umístění 7. parametru	Umístění 9. parametru

Pokud probíhá výpočet 2., 5., 7. a 9. parametru zvuku jako poslední krok před přechodem na následující foném vektoru fonémů, tak dojde po vypočítání parametrů zvuků ke změně umístění hodnot ve vektoru proměnných. Změna umístění hodnot v rámci vektoru proměnných je popsána v tabulce 2.13.

Úprava vektoru proměnných podle následujícího fonému

Úprava vektoru proměnných podle následujícího fonému začíná změnou umístění hodnot v rámci vektoru proměnných. Změna umístění hodnot v rámci vektoru proměnných je popsána v tabulce 2.14.

Po změně umístění hodnot ve vektoru proměnných je zkontrolováno, zda není aktuální foném posledním fonémem vektoru fonémů. Pokud jím je, tak následuje tvorba řídicích parametrů pro periodu generátoru šumu. Jinak je porovnán 2. bajt 8. hodnoty aktuálního fonému s 2. bajtem 8. hodnoty následujícího fonému.

Tab. 2.13: Změna umístění hodnot ve vektoru proměnných na konci úpravy řídicích parametrů aktuálního fonému

Původní pozice	Nová pozice
2. pozice	1. pozice
5. pozice	4. pozice
8. pozice	7. pozice
1. bajt 3. pozice	2. bajt 3. pozice
1. bajt 6. pozice	2. bajt 6. pozice
1. bajt 9. pozice	2. bajt 9. pozice
1. bajt 10. pozice	2. bajt 10. pozice
12. pozice	11. pozice

Tab. 2.14: Změna umístění hodnot ve vektoru proměnných při úpravě vektoru proměnných podle následujícího fonému

Původní pozice	Nová pozice
2. bajt 11. pozice	2. bajt 13. pozice
1. bajt 11. pozice	2. bajt 14. pozice
2. pozice	1. pozice
5. pozice	4. pozice
8. pozice	7. pozice
1. bajt 3. pozice	2. bajt 3. pozice
1. bajt 6. pozice	2. bajt 6. pozice
1. bajt 9. pozice	2. bajt 9. pozice
1. bajt 10. pozice	2. bajt 10. pozice

Pokud je 2. bajt 8. hodnoty aktuálního fonému menší než 2. bajt 8. hodnoty následujícího fonému, tak jsou do vektoru proměnných vloženy hodnoty podle 2. způsobu v tabulce 2.15. Po vložení vybraných hodnot aktuálního fonému do vektoru proměnných je prohozena hodnota umístění aktuálního a následujícího fonému.

Pokud 2. bajt 8. hodnoty aktuálního fonému je větší nebo roven 2. bajtu 8. hodnoty následujícího fonému, tak jsou do vektoru proměnných vloženy hodnoty podle 1. způsobu v tabulce 2.15. Po vložení vybraných hodnot následujícího fonému do vektoru proměnných nenásleduje prohození umístění aktuálního a následujícího fonému.

Po vložení hodnot do vektoru proměnných je porovnán počet zvuků aktuálního fonému s 2. bajtem 11. pozice vektoru proměnných.

Tab. 2.15: Vložení hodnot do vektoru proměnných podle 2. bajtu 8. hodnoty aktuálního a následujícího fonému

Umístění	Vložená hodnota
1. způsob	
2. bajt 11. pozice	2. bajt 10. hodnoty aktuálního fonému
2. bajt 12. pozice	1. bajt 9. hodnoty aktuálního fonému
1. bajt 11. pozice	1. bajt 15. hodnoty aktuálního fonému
1. bajt 12. pozice	2. bajt 15. hodnoty aktuálního fonému
2. způsob	
2. bajt 12. pozice	2. bajt 10. hodnoty následujícího fonému
2. bajt 11. pozice	1. bajt 9. hodnoty následujícího fonému
1. bajt 12. pozice	1. bajt 15. hodnoty následujícího fonému
1. bajt 11. pozice	2. bajt 15. hodnoty následujícího fonému

Když je počet zvuků menší, tak je počet zvuků aktuálního fonému vložen do 2. bajtu 11. pozice vektoru proměnných. Dále je porovnán počet zvuků aktuálního fonému s 1. bajtem 11. pozice, a pokud je menší, tak je vložen do dané pozice.

Následující část úpravy vektoru proměnných je podprogram, který je opakovaně volán, a který má 2 vstupní proměnné. Podprogram začíná kontrolou 2. proměnné. Pokud je 2. proměnná rovna 0, tak je do 1. proměnné vložena 0.

Když 2. proměnná není rovna 0, tak je od ní odečteno číslo 1. A když 2. proměnná není stále rovna 0, tak je podprogram ukončen beze změny 1. proměnné. Jestliže 2. proměnná je rovna 0, tak je hodnota 1. proměnné aritmeticky posunuta o 1 bit doprava.

Podprogram je volán maximálně sedmkrát a po návratu z podprogramu je k 1. proměnné přičtena vybraná hodnota aktuálního fonému. Výsledek je následně vložen na vybranou pozici vektoru proměnných.

Informace o vybrané hodnotě aktuálního fonému, vybrané pozici vektoru proměnných a vstupních proměnných jsou vypsány v tabulce 2.16. Volání podprogramu pro 1. řádek tabulky nastane, jenom pokud následující foném není fonémem R. Když jím je, tak je pouze vložena 3. hodnota aktuálního fonému na 2. pozici ve vektoru proměnných. Na obrázku 2.6 je zobrazen vývojový diagram úpravy vektoru proměnných podle následujícího fonému.

Tab. 2.16: Informace pro podprogram při úpravách vektoru proměnných podle následujícího fonému

	1. proměnná (následující foném)	2. proměnná (aktuální foném)	Hodnota aktuálního fonému	Pozice vektoru proměnných
1.	2. hodnota	1. bajt 8. hodnoty	3. hodnota	2. pozice
2.	4. hodnota	1. bajt 8. hodnoty	5. hodnota	5. pozice
3.	6. hodnota	2. bajt 9. hodnoty	7. hodnota	8. pozice
4.	1. bajt 10. hodnoty	1. bajt 14. hodnoty	2. bajt 11. hodnoty	1.bajt 3.pozice
5.	1. bajt 11. hodnoty	1. bajt 14. hodnoty	2. bajt 12. hodnoty	1.bajt 6.pozice
6.	1. bajt 12. hodnoty	1. bajt 14. hodnoty	2. bajt 13. hodnoty	1.bajt 9.pozice
7.	1. bajt 13. hodnoty	1. bajt 14. hodnoty	2. bajt 14. hodnoty	1.bajt 10.pozice

Úprava parametrů zvuku vektoru řídicích parametrů

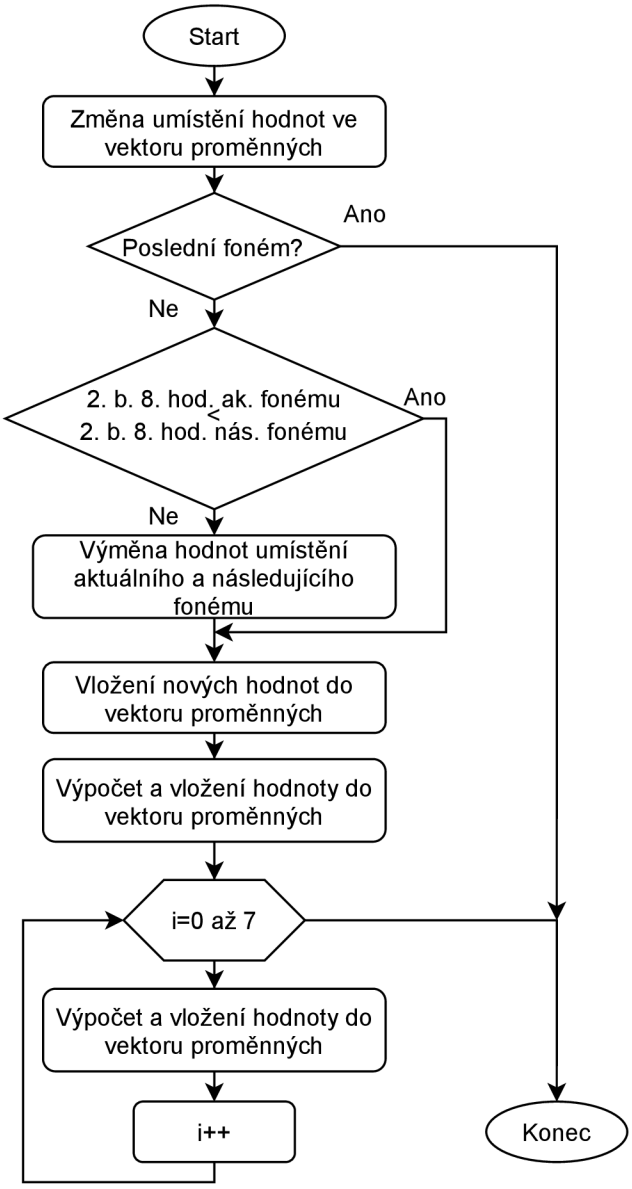
Úprava parametrů zvuku vektoru řídicích parametrů začíná odečtením hodnoty 2. bajtu 13. pozice a 2. bajtu 11. pozice vektoru proměnných od hodnoty počtu zvuků aktuálního fonému. Pokud je výsledek odčítání záporný nebo roven 0, tak nedochází k úpravě 4., 6, a 8. parametru zvuku vektoru řídicích parametrů.

Výsledná hodnota dále určuje pro kolik zvuků se má opakovat cyklus úpravy 4., 6. a 8. parametru zvuku. Dále je vypočítána hodnota umístění ve vektoru řídicích parametrů. Hodnota umístění je vypočítána vynásobením hodnoty 2. bajtu 13. pozice číslem 9.

Cyklus úpravy 4., 6. a 8. parametru zvuku začíná načtením hodnoty na 1. pozici ve vektoru proměnných, ke které je následně přičteno číslo 16 a výsledná hodnota je nakonec logicky posunuta o 5 bitů doprava. Po posunutí je výsledná hodnota vložena do 4. parametru aktuálního zvuku.

Stejným způsobem je upravena hodnota na 4. pozici vektoru proměnných a výsledná hodnota po posunutí je vložena do 6. parametru aktuálního zvuku. Jako poslední je opět stejným způsobem upravena hodnota ze 7. pozice a výsledná hodnota je pro změnu vložena do 8. parametru. Před přechodem na další zvuk vektoru řídicích proměnných je k hodnotě 2. bajtu 13. pozice přečteno číslo 1.

Po ukončení úpravy 4., 6. a 8. parametru zvuku jsou od hodnoty počtu zvuků aktuálního fonému odečteny hodnoty 2. bajtu 14. pozice a 1. bajtu 11. pozice. Pokud je výsledek záporný nebo roven 0, tak nedochází k úpravě 2., 5., 7. a 9. parametru zvuku vektoru řídicích parametrů. Výsledná hodnota dále určuje pro kolik zvuků se má opakovat cyklus úpravy 2., 5., 7. a 9. parametru zvuku.



Obr. 2.6: Vývojový diagram úpravy vektoru proměnných podle následujícího fonému

Před zahájením cyklu úpravy 2., 5., 7. a 9. parametru zvuku je vypočtena hodnota umístění ve vektoru řídicích parametrů. Hodnota umístění je získána vynásobením hodnoty 2. bajtu 14. pozice vektoru proměnných číslem 9.

Cyklus úpravy 2., 5., 7. a 9. parametru zvuku začíná vynásobením hodnoty 2. bajtu 3. pozice vektoru proměnných číslem 3 a následným odečtením čísla 89. Pokud je výsledek odčítání kladný, tak je logicky posunut o 2 bity doprava a následně je uložen do 5. parametru aktuálního zvuku. Když výsledek odčítání není kladný, tak je do 5. parametru vložena 0.

Stejným způsobem jsou získány i hodnoty pro vložení do 7., 9. a 2. parametru aktuálního zvuku. Přehled hodnot vektoru proměnných pro úpravu 2., 5., 7. a 9. parametru aktuálního zvuku je v tabulce 2.17.

Tab. 2.17: Přehled hodnot vektoru proměnných pro úpravu 2., 5., 7. a 9. parametru aktuálního zvuku

Parametr	Pozice ve vektoru proměnných
2. parametr	2. bajt 10. pozice
5. parametr	2. bajt 3. pozice
7. parametr	2. bajt 6. pozice
9. parametr	2. bajt 9. pozice

Před přechodem na další zvuk vektoru řídicích parametrů je k hodnotě 2. bajtu 14. pozice přičteno číslo 1. Na obrázku B.7 je zobrazen vývojový diagram úpravy parametrů zvuku vektoru řídicích parametrů.

Po úpravě parametrů zvuku řídicích parametrů opět následuje výpočet 4., 6. a 8. parametru zvuku vektoru řídicích parametrů a výpočet 2., 5., 7. a 9. parametru vektoru řídicích parametrů.

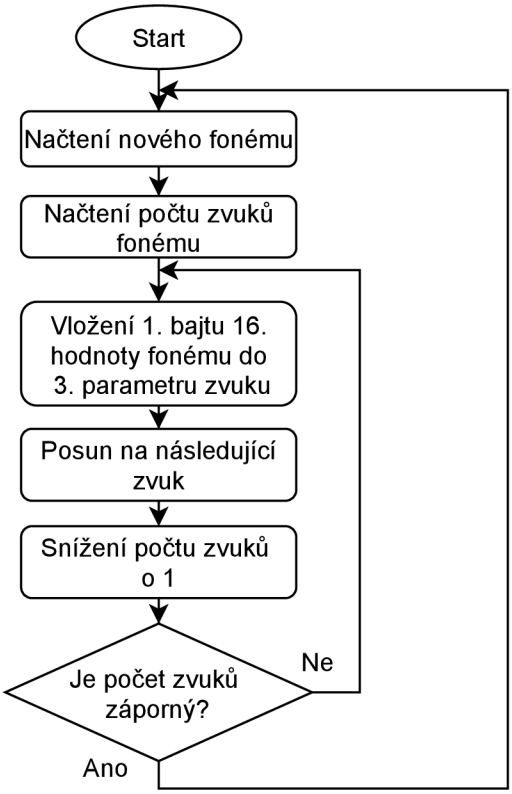
Tvorba řídicích parametrů pro periodu generátoru šumu

Při tvorbě řídicích parametrů pro periodu generátoru šumu je do 3. parametru zvuku vektoru řídicích parametrů vložena hodnota 1. bajtu 16. hodnoty aktuálního fonému v tabulce fonémů. Počet následujících zvuků, do kterých má být vložena hodnota 1. bajtu 16. hodnoty aktuálního fonému, určuje počet zvuků aktuálního fonému zmenšený o 1. Vývojový diagram je zobrazen na obrázku 2.7.

Tvorba řídicích parametrů pro změnu způsobu výpočtu u znělých zvuků

Tvorba řídicích parametrů pro změnu způsobu výpočtu u znělých zvuků začíná odečtením čísla 1 od hodnoty počtu zvuků aktuálního fonému vektoru fonémů. Pokud je výsledek odčítání záporný, tak se přejde na následující foném. Výsledek dále určuje do kolika následujících zvuků má být vložena hodnota proměnné změna. Proměnná změna má výchozí hodnotu 66.

Následně je ověřeno zda je aktuální foném mezerou. Pokud jí je, tak následuje rovnou zápis hodnot do vektoru řídicích parametrů. Při zápisu hodnot je jako první vložena hodnota proměnné změna do 1. parametru aktuálního zvuku. Proměnná změna je následně porovnána s číslem 66, a pokud je větší, tak je od proměnné změna odečteno číslo 1.



Obr. 2.7: Vývojový diagram tvorby řídicích parametrů pro periodu generátoru šumu

Když je menší, tak je k proměnné změna přičteno číslo 1. Potom následuje zápis hodnot do vektoru řídicích parametrů pro další zvuk.

Když aktuální foném není mezerou, ale je speciálním fonémem, tak je hodnota posunu nazpět ve vektoru řídicích parametrů nastavena na hodnotu -20. Následně je ověřen předchozí foném vektoru fonémů. Pokud předchozí foném není speciálním fonémem, tak je hodnota jeho počtu zvuků přičtena k hodnotě posunu nazpět ve vektoru řídicích parametrů. Když je výsledná hodnota posunu nazpět záporná, tak je ověřen další předchozí foném. Vektorem fonémů se postupuje směrem zpět dokud je hodnota posunu nazpět záporná, není nalezen speciální foném nebo není ověřen 1. foném vektoru fonémů. Pokud je postup vektorem fonémů ukončen kvůli nezáporné hodnotě posunu nazpět, tak je hodnota posunu nastavena na 0.

Po ukončení procházení vektoru fonémů je k hodnotě posunu nazpět přičteno číslo 20. Pokud je výsledná hodnota posunu nazpět rovna 0, tak je do proměnné změna vložena hodnota 66 a následuje zápis hodnot do vektoru řídicích parametrů. Když výsledná hodnota posunu nazpět není rovna 0, tak je kontrolováno jestli není aktuální foném speciálním fonémem tečka.

Jestli jím je, tak je do proměnné úprava vloženo číslo 1. Pokud aktuální foném není speciálním fonémem tečka, ale je speciálním fonémem ?, tak je do proměnné úprava vloženo číslo -1. A pokud není ani speciálním fonémem ?, tak je vloženo číslo 1.

Hodnota posunu nazpět může být dále upravena dvěma způsoby. První způsob úpravy nastane, pokud aktuální foném není speciálním fonémem tečka ani ?. Při tomto způsobu je hodnota posunu nazpět logicky posunuta o 1 bit doprava a následně je k ní přičteno číslo 1. Dále je hodnota posunu nazpět vynásobena číslem -9. Druhý způsob úpravy nastane, pokud je aktuální foném buď speciálním fonémem tečka nebo ?. Při 2. způsobu je hodnota posunu nazpět pouze vynásobena číslem -9.

Proměnná úprava je přičtena k 1. parametru zvuku, který je určen hodnotou posunu nazpět ve vektoru řídicích parametrů. Po zapsaní hodnoty do vektoru řídicích parametrů je k hodnotě posunu nazpět přičteno číslo 9 a následuje uložení proměnné úprava na novou pozici ve vektoru řídicích parametrů. Před uložením proměnné úprava na novou pozici je proměnná zvětšena nebo zmenšena o 1 podle toho jestli její počáteční hodnota byla -1 nebo 1. Ukládání do vektoru řídicích parametrů se opakuje dokud je hodnota posunu nazpět záporná. Nakonec se do proměnné změna vloží hodnota 66 a následuje zápis hodnot do vektoru řídicích parametrů.

Pokud aktuální foném není speciálním fonémem, tak je ověřen jeho tón a délka. Když je nastaven, alespoň 1 z těchto parametrů, tak je vypočítána nová hodnota proměnné změna. Hodnota proměnné změna je vypočítána vynásobením 3. bajtu aktuálního fonému číslem 3, následným negováním výsledku a přičtením čísla 66. Takto vypočítaná hodnota je vložena do 1. parametru aktuálního zvuku a do všech následujících zvuků, jejichž počet je určen odečtením čísla 1 od hodnoty počtu zvuků aktuálního fonému. Pokud nemá aktuální foném nastaven tón ani délku, následuje zápis hodnot do vektoru řídicích parametrů.

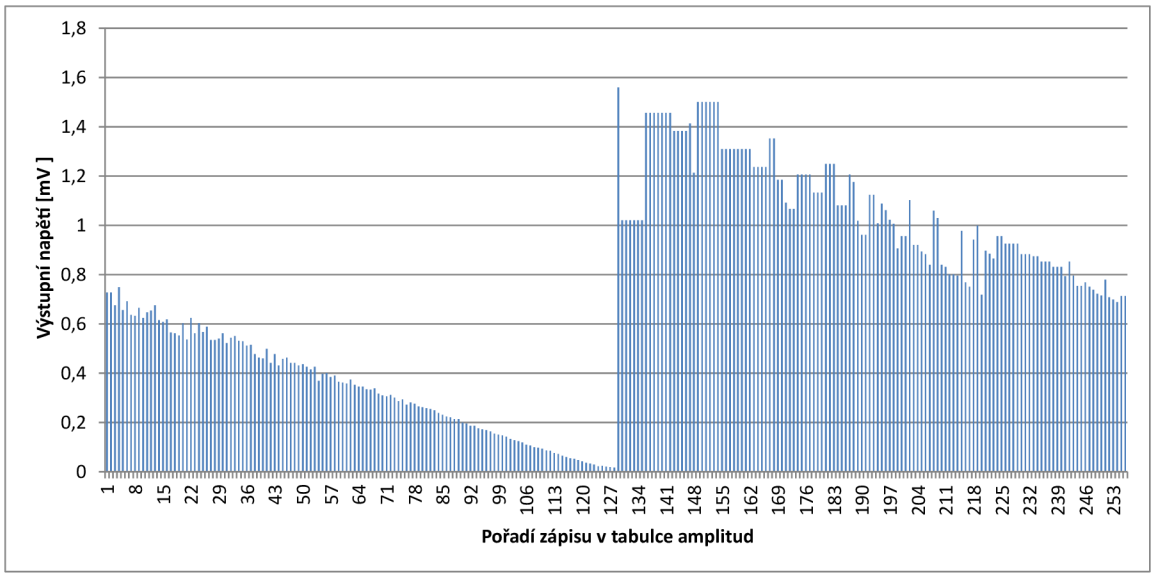
Po zapsání hodnot řídicích parametrů je algoritmus opakován pro všechny fonémy ve vektoru fonémů. Vývojový diagram tvorby řídicích parametrů pro změnu způsobu výpočtu u znělých zvuků je zobrazen na obrázku B.8.

2.3.7 Generování řeči

Pro generování řeči je použita pulzně kódová modulace, při které je analogový signál vytvářen převodem binárních hodnot na odpovídající úroveň napětí v pravidelných intervalech [19].

Analogový signál je na výstupních kanálech zvukového obvodu AY-3-8910 tvořen nastavením jejich amplitud na požadovanou hodnotu v rozsahu 0 až 15 a odpojením generátorů tónu.

Kombinací amplitud 3 výstupních kanálů zvukového obvodu lze vytvořit 256 úrovní výstupního napětí [20]. Graf 256 hodnot výstupního napětí je zobrazena na obrázku 2.8.



Obr. 2.8: Graf 256 hodnot výstupního napětí

Generování řeči začíná výpočtem tempa z hodnoty nastaveného tempa řeči a intonace řeči. Tempo je vypočítáno vynásobením nastaveného tempa řeči číslem 77 a následným vydělením hodnotou nastavené intonace řeči. Po výpočtu tempa následuje inicializace časovače A víceúčelové periferie MC68901, pomocí kterého je v pravidelných intervalech voláno přerušení pro generování řeči. Časovač A je nastaven do režimu zpoždění a má předděličku nastavenu na dělení 4. Pro nastavení frekvence, s jakou má nastat přerušení pro generování řeči je použita hodnota intonace řeči. Při generování řeči se střídá zápis amplitud pro znělé a neznělé zvuky.

Pro přechod na zápis nového zvuku je použita vypočtená hodnota tempa, která je při každém zápisu amplitud snížena o 1 a při jejímž snížení na zápornou hodnotu dojde k přechodu na zápis nového zvuku. Hodnota tempa je následně vrácena na původní vypočítanou hodnotu. O přechodu mezi zápisem znělého a neznělého zvuku rozhoduje 2. parametr zvuku vektoru řídicích parametrů. V tomto parametru je uloženo nastavení amplitudy výstupního kanálu C v případě, že jde o neznělý zvuk. Pokud to není neznělý zvuk, tak je parametr roven 0 a zápis amplitud neznělého zvuku přejde na zápis amplitud zvuku znělého.

Při zápisu znělého zvuku jsou 2 druhy výpočtu umístění hodnot v tabulce amplitud. První způsob je použit, pokud je 1. parametr zvuku kladný.

Druhý způsob je použit, pokud není a po jeho provedení je nastavena původní hodnota 1. parametru. První parametr je stejně jako hodnota tempa při každém zápisu amplitud aktuálního zvuku snížen o 1.

Při 1. způsobu výpočtu umístění v tabulce amplitud je použit 4. až 9. parametr aktuálního zvuku. 4., 6. a 8. parametr je přičítán k 1. bajtu hodnoty umístění v tabulce amplituda_offset a 5., 7., a 9. parametr mění 2. bajt hodnoty umístění v tabulce amplituda_offset. První bajt hodnoty umístění v tabulce amplituda_offset má výchozí hodnotu 0. 4. a 5. parametr tvoří umístění 1. hodnoty, 6. a 7. parametr tvoří umístění 2. hodnoty a umístění poslední hodnoty tvoří 8. a 9. parametr. Hodnoty nalezené v tabulce amplituda_offset jsou sečteny a následně je proveden logický součin vypočítané hodnoty s hodnotou 255. Nakonec je vypočítaná hodnota logicky posunuta o 3 bity doleva. S její pomocí je nalezen začátek jednoho zápisu amplitud v tabulce amplitud.

Druhý způsob výpočtu je stejný jako 1. způsob, jenom je 1. bajt hodnoty umístění v tabulce amplituda_offset před přičtením parametru nastaven zpět do 0. Po nalezení začátku zápisu amplitud v tabulce amplitud následuje zápis do registrů amplitud 3 výstupních kanálů zvukového obvodu, které jsou všechny včetně generátoru šumu vypnuty.

Při zápisu neznělého zvuku je nejprve nastaven generátor šumu, jehož perioda je uložena ve 3. parametru zvuku. Dále je nastavena amplituda výstupního kanálu C na hodnotu, která je uložena v 2. parametru zvuku. Po nastavení generátoru šumu a amplitudy výstupního kanálu C je vypočten začátek zápisu amplitud v tabulce amplitud.

Postup výpočtu je stejný jako 1. způsob výpočtu pro znělé zvuky. Při zápisu neznělého zvuku jsou nastaveny pouze amplitudy pro výstupní kanály A a B. Vývojový diagram generování řeči je zobrazen na obrázku B.9.

2.3.8 Ukončení generování řeči

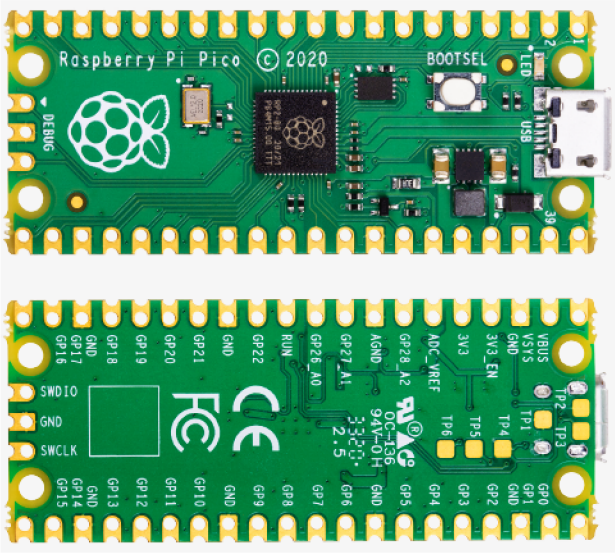
Ukončení generování řeči začíná vynulováním bitu 5 registru čekajících přerušení, registru masky přerušení a registru přerušení v provozu víceúčelové periferie MC68901. Následně jsou opět vypnuty výstupní kanály zvukového obvodu AY-3-8910 a povoleno přerušení časovače C periferie MC68901 s frekvencí 200 MHz.

3 Vývojová deska Raspberry Pi Pico

Jako vhodný embedded systém byla vybrána vývojová deska Raspberry Pi Pico od společnosti Raspberry Pi, která byla uvedena na trh v roce 2021 a má rozměry 51 x 21 x 1 mm [21].

Vývojová deska obsahuje mikrontrolér Raspberry Pi RP2040, který má provozní napětí 3,3 V. Provozní napětí 3,3 V je tvořeno pomocí spínaného regulátoru RT6150. Pro správnou funkci mikrokontroléru RP2040, vývojová deska obsahuje taky 2 MB paměti typu flash a 12 MHz krystalový oscilátor [21] [22].

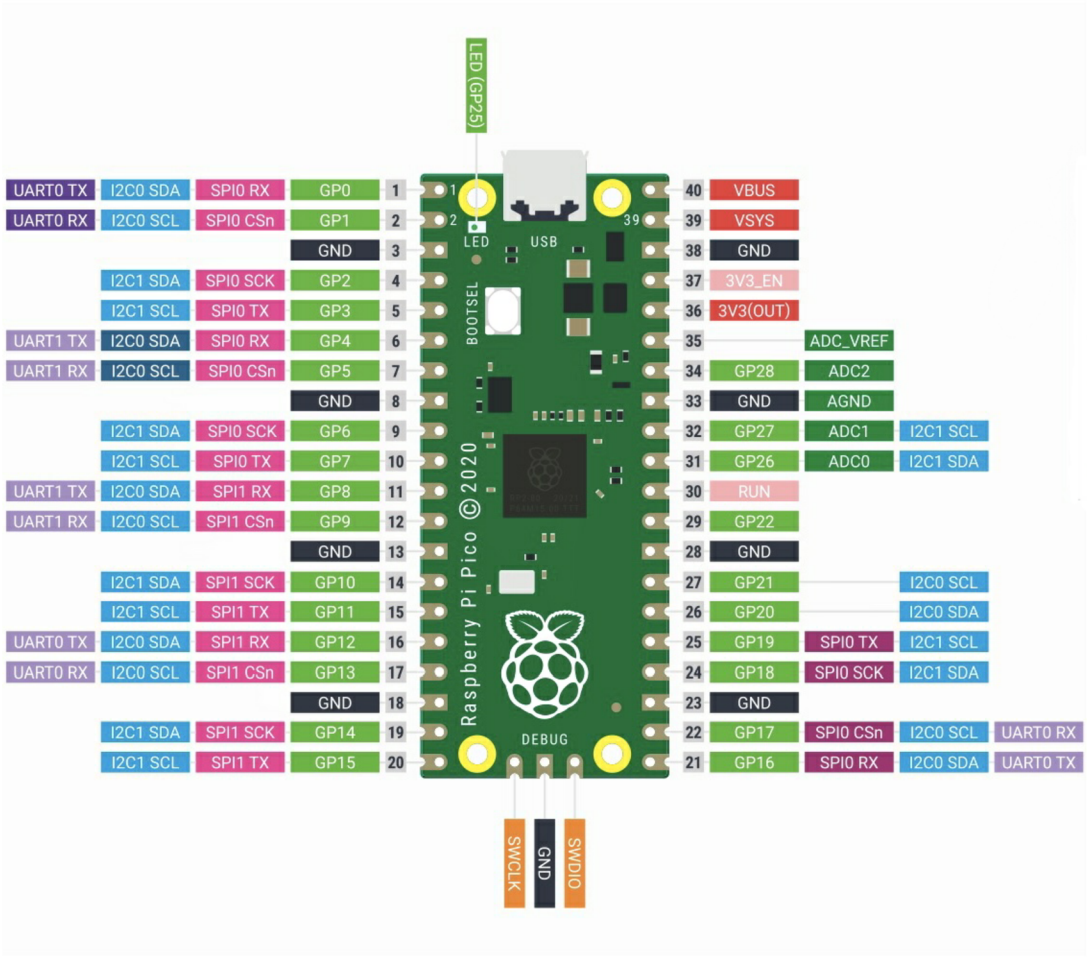
Dále obsahuje 26 víceúčelových vstupně/výstupních pinů GPIO, z nichž 3 lze použít jako analogové piny. Těchto 26 pinů je připojeno k 26 odpovídajícím pinům RP2040 o napětí 3,3 V. Na obrázku 3.2 jsou tyto piny označeny GP0 až GP22 a GP26 až GP28 [21].



Obr. 3.1: Vývojová deska Raspberry Pi Pico[21]

Raspberry Pi Pico lze napájet pomocí Micro-USB B konektoru, který je možné kromě napájení použít i pro přenos dat a nahraní programu do vývojové desky. Dále je možné vývojovou desku napájet připojením externího zdroje napájení k pinu VSYS. Doporučený rozsah napětí externího zdroje napájení je 1,8 V až 5,5 V a vstupní napětí Micro-USB B konektoru je 5 V [21].

Program lze do vývojové desky nahrát 2 způsoby. Prvním způsobem je nahrání programu do desky pomocí Micro-USB B konektoru. V tomto případě přejde vývojová deska do režimu velkokapacitního paměťového USB zařízení držením tlačítka BOOTSEL při připojování vývojové desky k PC [21].



Obr. 3.2: Rozložení pinů vývojové desky Raspberry Pi Pico[21]

Druhým způsobem je použití rozhraní pro sériovou diagnostiku, neboli zkráceně SWD. Toto rozhraní lze dále použít pro ladění nahraného programu [21].

3.1 Mikrokontrolér Raspberry Pi RP2040

Mikrokontrolér Raspebrry Pi RP2040 obsahuje dvoujádrový procesor Arm Cortex-M0+ s maximální frekvencí 133 MHz a integrovanou SRAM paměť o velikosti 264 kB, která je rozdělena do 6 nezávislých buněk. Kromě integrované SRAM paměti mikrokontrolér obsahuje 16 kB ROM paměti, ve které je uložena například úvodní spouštěcí procedura mikrokontroléru, zaváděcí sekvence paměti typu flash nebo zavaděč programu [22].

Dále mikrokontrolér obsahuje 30 víceúčelových vstupně/výstupních pinů GPIO, z nichž 4 lze použít jako analogové piny [22].

Z periférií mikrokotrolér podporuje 2x sériovou komunikaci UART, 2x komunici SPI, 2x komunikaci I2C, 8x generátor PWM a 8x PIO stavový automat. Každý generátor PWM má 2 nezávislé výstupní kanály, 16bitový čítač, režim citlivý na hranu vstupního signálu pro měření frekvence a režim citlivý na úroveň vstupního signálu pro měření střídy signálu [22].

V rámci RP2040 je k dispozici 1 64bitový časovač, pro který lze nastavit až 4 alarmy. Dále je pro měření času možné použít i 16bitových čítačů 8 generátorů PWM nebo hodin reálného času RTC [22].

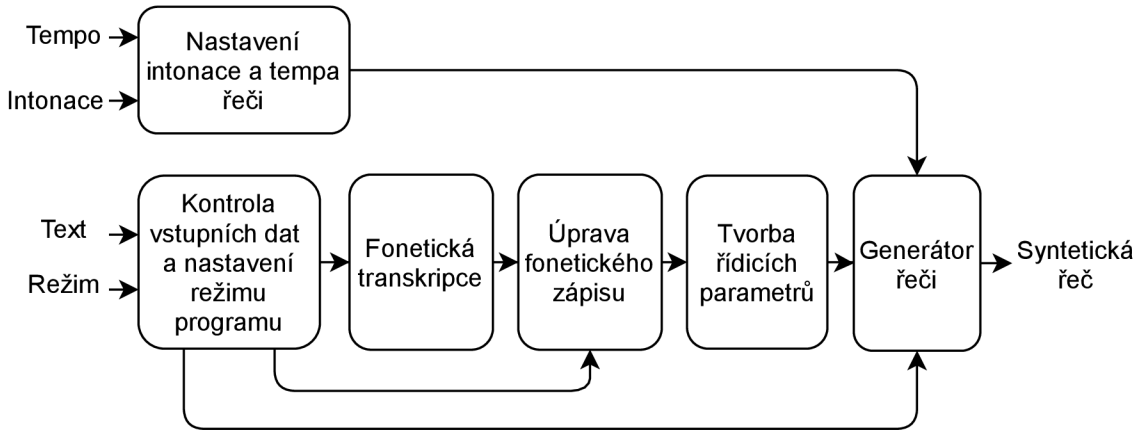
3.2 Důvody volby a omezení vývojové desky Raspberry Pi Pico

Raspberry Pi Pico je cenově dostupnou vývojovou deskou, která podporuje programovací jazyk C/C++. Je ji možné rozšířit o různé obvody jako jsou například reproduktory. Dále splňuje hardwarové požadavky potřebné pro generování audia, kterými jsou například časovače s vysokou frekvencí časování nebo dostatečně velká paměť pro uložení programu a jeho dat(řádově stovky kB) [21] [22].

Omezením vývojové desky může být absence pinu s výstupním napětím 5 V, což může komplikovat připojení obvodů s pracovním napětím 5 V [21].

4 Návrh hlasového generátoru a implementace jeho základních komponent

V následujícím textu je popsán návrh hlasového generátoru a implementace jeho základních komponent. Blokové schéma navrženého hlasového generátoru je zobrazeno na obrázku 4.1. Předlohou pro hlasový generátor byla architektura programu Atari 520ST Speech Synthesizer V2.0, která byla popsána v kapitole 2.



Obr. 4.1: Blokové schéma hlasového generátoru

Blokové schéma hlasového generátoru je složeno z 6 částí a 2 větví, které jsou spojeny v komponentě generátoru řeči. Vstupem do 1. větve jsou hodnoty požadovaného tempa a intonace řeči, které jsou následně v komponentě nastavení intonace a tempa řeči ověřeny a zpracovány. Intonace a tempo řeči jsou nastaveny samostatně, aby uživatel nemusel opakovaně zadávat jejich hodnoty i v případě, že nedochází k jejich změně.

Druhá větev blokového schématu tvoří hlavní algoritmus hlasového generátoru, jehož vstupními hodnotami je text a režim, ve kterém má hlavním algoritmus proběhnout. Hlavní algoritmus může proběhnout 4 způsoby, o jejichž výběru rozhoduje nastavený režim programu. Více bude popsáno v rámci následující podkapitoly 4.1.

4.1 Implementace základních komponent hlasového generátoru

Základní komponenty hlasového generátoru jsou napsány v jazyce C/C++ a jejich implementace je uvedena v následujícím textu.

4.1.1 Nastavení intonace a tempa řeči

Komponentu nastavení intonace a tempa řeči tvoří funkce s názvem TempoIntonace. Jako první je ve funkci ověřeno, zda jsou vstupní proměnné TempoIn a IntonaceIn v povoleném rozsahu 20 až 199. Pokud jsou v povoleném rozsahu, tak je uložena hodnota vstupní proměnné TempoIn do globální proměnné LastTempoIn a hodnota vstupní proměnné IntonaceIn je uložena do globální proměnné Intonace. Potom je vypočítána nová hodnota tempa, která je vypočítána vynásobením hodnoty globální proměnné LastTempoIn číslem 77 a následným vydělením globální proměnnou Intonace. Výsledek dělení je uložen do globální proměnné Tempo.

Pokud by uživatel chtěl změnit například jenom hodnotu intonace, tak vstupní hodnotu TempoIn nastaví na 0. Když je vstupní hodnota TempoIn rovna 0, nedojde k uložení vstupní proměnné TempoIn do globální proměnné LastTempoIn, čímž dojde k výpočtu nové hodnoty tempa z naposledy vložené hodnoty tempa. Pokud je vstupní proměnná IntonaceIn rovna 0, tak hodnota proměnné IntonaceIn není vložena do globální proměnné Intonace a nová hodnota globální proměnné Tempo je vypočtena z naposledy uložené hodnoty intonace.

Když nejsou vstupních proměnné v povoleném rozsahu, tak k žádné změně globálních proměnných nedojde. Globální proměnné LasTempoIn a Tempo mají výchozí hodnotu 79. Výchozí hodnota globální proměnné Intonace je 77.

4.1.2 Kontrola vstupních dat a nastavení režimu programu

Kontrolu vstupních dat a nastavení režimu programu tvoří funkce RezimVstup. Tato funkce má 2 vstupní proměnné, kterými jsou proměnná RezimIn a ukazatel na vstupní řetězec TextIn. Jako první je ověřena velikost vstupního řetěze pomocí funkce strlen() z knihovny string, která vrací délku vstupního řetězce. Pokud je délka vstupního řetězce větší než 256, což je maximální povolený počet znaků, nedojde k žádné změně a návratová hodnota funkce je 0.

Pokud je vstupní řetězec v rozsahu 1 až 256 a vstupní proměnná RezimIn má hodnotu 0 nebo 1, tak dojde k uložení ukazatele na vstupní řetězec do globální proměnné Text. V případě, že je vstupní proměnná RezimIn rovna hodnotě 2, tak je pouze uložena hodnota vstupní proměnné RezimIn do globální proměnné Rezim.

A když je vstupní proměnná RezimIn rovna hodnotě 3, tak dojde k otestování, zda hlasový generátor generuje řeč. K tomuto ověření dojde kontrolou hodnoty globální proměnné Generovani, která je během generování řeči rovna hodnotě 1. V tomto případě je funkce ukončena s návratovou hodnotou 2. Přehled režimů hlasového generátoru je zobrazen v tabulce 4.1.

Tab. 4.1: Přehled režimů hlasového generátoru

Režim	Popis
0	Generování řeči ze vstupního anglického textu
1	Generování řeči ze vstupního fonetického zápisu
2	Generování řeči z předchozích vstupních dat
3	Testování ukončení generování řeči

4.1.3 Fonetická transkripce

V komponentě Fonetická transkripce je převeden vstupní anglický text na fonetický zápis v případě, že je nastaven režim 0. Algoritmus převodu textu na fonetický zápis je rozdělen do 2 částí. První část algoritmu tvoří funkce TextUprava, ve které je vstupní řetězec vyčištěn od nežádoucích ASCII znaků. Pro vyčištění nežádoucích ASCII znaků je použita tabulka ZnakTyp. Hodnoty v tabulce určují do jaké kategorie daný ASCII znak spadá. Hodnoty a kategorie jsou stejné jako v tabulce 2.1 v programu Atari 520ST Speech Synthesizer V2.0.

Před ověřením kategorie ASCII znaku je každý ASCII znak porovnán s hodnotou 96 a pokud je větší nebo roven této hodnotě, dojde k odečtení čísla 32 od hodnoty daného ASCII znaku. Následně je nalezena odpovídající hodnota v tabulce ZnakTyp, jejíž pozice je určena hodnotou daného ASCII znaku. Pokud je odpovídající hodnota v tabulce ZnakTyp rovna 0, dojde k vynechání daného ASCII znaku. ASCII znaky, které nemají odpovídající hodnotu v tabulce ZnakTyp rovnu 0, jsou uloženy do globálního pole TextUpraveny. Jako poslední je před 1. prvek a za poslední prvek globálního pole TextUpraveny vložena mezera, která je tam vložena kvůli následující části převodu textu na fonetický zápis.

Druhou část algoritmu tvoří 5 funkcí, které budou popsány ve vlastních podkapitolách. Hlavní funkcí je funkce s názvem FonemTransAlg, která tvoří hlavní algoritmus převodu vyčištěného textu na fonetický zápis. Dalšími funkcemi jsou funkce FonemTransNerovno, FonemTransRovno, PrefixSufix a PrefixSufixKontrola. Pro převod textu na fonetických zápis je dále důležitá tabulka FoneticTrans, ve které jsou uloženy části slov, speciální znaky, podmínky pro použití jejich fonetických zápisů a nakonec jejich fonetické zápisy. Informace pro 1 část slova nebo speciálního znaku budou dohromady nazývány část slova. Na konci každé části slova je hodnota 0, která značí její konec. Pro pohyb v tabulce FoneticTrans je použita proměnná PolohaTab.

Funkce FonemTransAlg

Algoritmus funkce FonemTransAlg je opakován pro všechny ASCII znaky v globálním poli TextUpraveny.

Pokud je aktuálním ASCII znakem v globálním poli TextUpraveny speciální znak nebo číslo, tak začíná průchod tabulkou FoneticTrans od pozice 4392, kde začíná část tabulky obsahující všechny speciální znaky a čísla. Pokud je aktuální ASCII znak v globálním poli TextUpraveny písmenem, tak začíná průchod tabulkou v části pro dané písmeno. Poloha dané části tabulky je dána hodnotou v tabulce Poloha. Umístění aktuálního ASCII znaku v globálním poli TextUpraveny je uloženo v proměnné Aktual a hodnota proměnné Aktual je uložena do proměnné AktualPrefixSufix.

Samotné hledání odpovídající části slova v tabulce FoneticTrans začíná načtením 1. prvku aktuální části slova a nastavením jeho bitu 7 do 0. Následně je tato hodnota porovnána s ASCII znakem, který je určený proměnnou Aktual. Pokud se nerovnají tak, je volána funkce FonemTransNerovno, jejíž návratová hodnota je vložena do proměnné PolohaTab, a do proměnné AktualPrefixSufix je opět uložena hodnota proměnné Aktual. Potom opět následuje kontrola 1. prvku aktuální části slova. V případě, že se rovnají, tak následuje ověření neupravené hodnoty 1. prvku aktuální části slova.

Pokud je hodnota 1. prvku aktuální části slova před nastavením jeho bitu 7 větší než 0, tak následuje přechod na další prvek aktuální části slova a proměnná AktualPrefixSufix je zvýšena o 1. Potom následuje návrat na ověření upravené hodnoty 1. prvku aktuální části slova jenom není ověřován 1. prvek, ale prvek následující.

Když je hodnota neupraveného 1. prvku aktuální části slova záporná nebo rovna 0, tak následuje přechod na další prvek aktuální části slova a do proměnné AktualPrefix je uložena hodnota proměnné Aktual.

Následně je volána funkce PrefixSufix v režimu ověřování předchozího ASCII znaku v globálním poli TextUpraveny. Pokud má návratovou hodnotu 1, tak následuje ověření následujícího ASCII znaku v globálním poli TextUpraveny. V případě návratové hodnoty 0 následuje volání funkce FonemTransNerovno a návrat na ověření upraveného 1. prvku aktuální části slova. A pokud je návratová hodnota rovna 2, tak došlo k chybě a převod textu na fonetický zápis je ukončen s návratovou hodnotou 0.

Ověření následujícího ASCII znaku v globálním poli TextUpraveny je provedeno opět pomocí funkce PrefixSufix tentokrát, ale v režimu ověřování následujícího ASCII znaku. V případě návratových hodnot 0 a 2 následuje stejný postup jako u předchozího volání funkce PrefixSufix. V případě návratové hodnoty 1 dojde k volání funkce FonemTransRovno, protože byla nalezena odpovídající část slova.

Po návratu z funkce FonemTransRovno je proměnná AktualPrefixSufix zvětšena o 1 a následně je tato hodnota vložena do proměnné Aktual. Dále následuje návrat na začátek funkce FonemTransAlg.

Když je ověřena poslední část slova v tabulce FoneticTrans, tak nastane ukončení fonetické transkripce s návratovou hodnotou 0. Na obrázku C.1 je zobrazen vývojový diagram funkce FonemTransAlg.

Funkce PrefixSufix

Funkce PrefixSufix tvoří hlavní část ověření ASCII znaků, které leží před nebo za aktuálním ASCII znakem v globálním poli TextUpraveny. Vstupními proměnnými jsou reference Poloha, která určuje polohu v globálním poli TextUpraveny, reference PolohaTab, která určuje polohu v tabulce FoneticTrans, a poslední proměnou je proměnná Sufix, která určuje v jakém režimu má funkce proběhnout. Když má proměnná hodnotu 0, tak je nastavený režim ověřování předchozího ASCII znaku v globálním poli TextUpraveny. A když má proměnná hodnotu 1, tak je nastavený režim ověřování následujícího ASCII znaku.

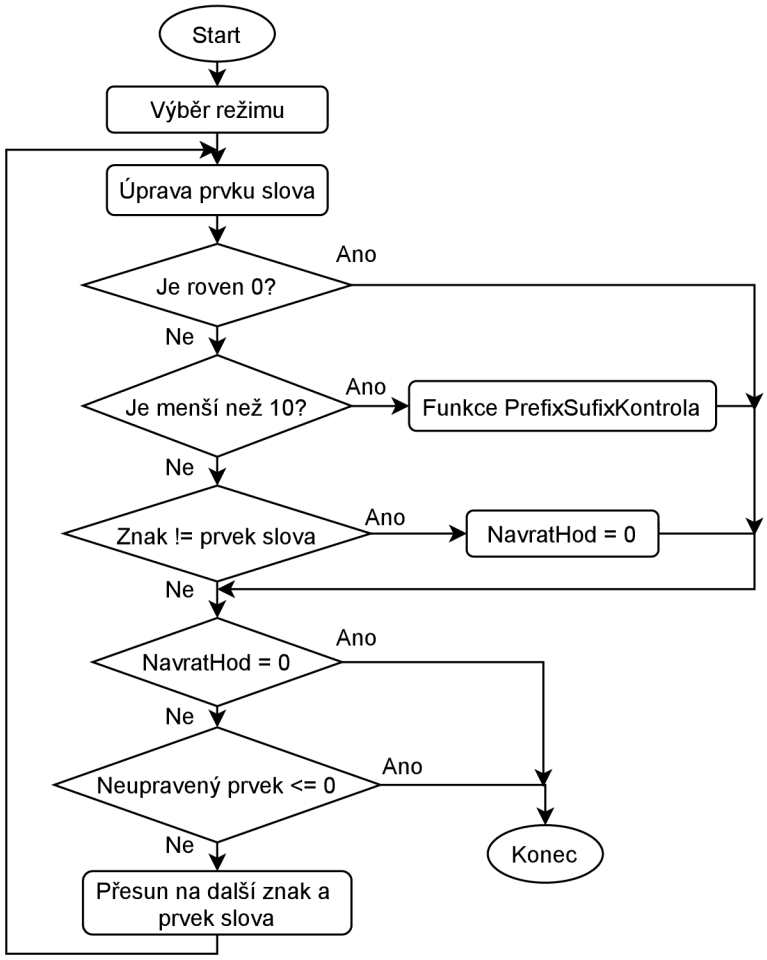
Algoritmus funkce PrefixSufix začíná načtením aktuálního prvku části slova v tabulce FoneticTrans, který je následně vložen do proměnné ZnakTab. V proměnné ZnakTab je potom nastaven bit 7 do 0. Pokud je hodnota proměnné ZnakTab rovna 0, tak je zvětšena hodnota proměnné PolohaTab o 1 a následuje načtení dalšího prvku aktuální části slova.

Pokud je hodnota proměnné ZnakTab větší než 9, tak je proměnná Poloha zmenšena o 1 v případě režimu ověření předchozího ASCII znaku. V případě režimu ověření následujícího ASCII znaku je k proměnné Poloha přičtena 1. Následně je ověřeno, že je ASCII znak, na který ukazuje proměnná Poloha roven proměnné ZnakTab. Pokud ASCII znak není roven proměnné ZnakTab, tak dojde k ukončení funkce PrefixSufix s návratovou hodnotou 0. Jinak následuje zvětšení hodnoty proměnné PolohaTab o 1 a načtení dalšího prvku aktuální části slova.

V případě, že je proměnná ZnakTab v rozsahu 1 až 9 dojde k volání funkce PrefixSufixKontrola, ve které jsou ověřeny předchozí nebo následující ASCII znaky v globálním poli TextUpraveny podle toho o jaký druh ASCII znaku jde. Pokud je návratová hodnota funkce PrefixSufixKontrola rovna 0, tak dojde k ukončení funkce PrefixSufix s návratovou hodnotou 0. V případě, že je návratovou hodnotou číslo 2, tak dojde k ukončení funkce PrefixSufix s návratovou hodnotou 2. A pokud je návratová hodnota 1, tak je hodnota proměnné PolohaTab zvětšena o 1 a následuje ověřování následujícího prvku aktuální části slova.

Ověření prvků aktuální části slova probíhá dokud není neupravená hodnota aktuálního prvku menší než 0.

V tomto případě je funkce ukončena s návratovou hodnotou 1. Nebo je ověření aktuální části slova ukončeno z důvodu, že proměnná Poloha ukazuje na 1. nebo poslední prvek v globálním poli TextUpraveny. V tomto případě je funkce ukončena s návratovou hodnotou 0. Vývojový diagram funkce PrefixSufix je zobrazen na obrázku 4.2.



Obr. 4.2: Vývojový diagram funkce PrefixSufix

Funkce PrefixSufixKontrola

Funkce PrefixSufixKontrola ověřuje předchozí nebo následující ASCII znak v globálním poli TextUpraveny podle toho do jaké katagorie ASCII znaků má daný ASCII znak patřit. O tom jaká kategorie ASCII znaků je ověřena rozhoduje hodnota ve vstupní proměnné ZnakTab.

V případě kontroly předchozích ASCII znaků může proměnná ZnakTab dosahovat až hodnoty 8 a v případě kontroly následujících ASCII znaků může dosahovat hodnoty až 9. Pokud při ověřování předchozích ASCII znaků má proměnná ZnakTab hodnotu 9 dojde k ukončení funkce s návratovou hodnotou 2.

Algoritmus začíná odečtením 1 od vstupní proměnné reference Poloha podle, které je následně načten odpovídající ASCII znak v globálním poli TextUpraveny. V případě, že je prováděno ověřování následujícího ASCII znaku, je k proměnné Poloha naopak přičtena 1. Načtený ASCII znak je uložen do proměnné Znak. Podle proměnné Znak je nalezena odpovídající hodnota v tabulce ZnakTyp, která je uložena do proměnné TestHod.

Hodnota proměnné ZnakTab určuje, který bit proměnné TestHod je kontrolován. V tabulce 4.2 je vypsáno, které bity proměnné TestHod jsou kontrolovány při jaké hodnotě proměnné ZnakTab a jaká kategorie ASCII znaků je při tom ověřena.

Pokud má proměnná ZnakTab hodnotu 5, tak je nejprve ověřeno jestli je ASCII znak jedním z ASCII znaků C, X, G, J, Z, S a pokud jím je, tak je vrácena hodnota 1. Když aktuální ASCII znak není jedním z výše zmíněných ASCII znaků, ale je ASCII znakem H, tak je od proměnné Poloha odečtena 1 v případě ověřování předchozích ASCII znaků a v případě ověřování následujících ASCII znaků je k proměnné Poloha přičtena 1. A pokud ASCII znak, na který nově ukazuje proměnná Poloha, je ASCII znakem C nebo S, tak je vrácena hodnota 1. Jinak je vrácena hodnota 0.

Tab. 4.2: Přehled kategorií při ověřování ASCII znaků ve fonetické transkripci

ZnakTab	Ověřovaný bit	Kategorie ASCII znaků
1	0	Číslo, speciální znak
2	1	Samohlásky
3	2	Souhláska
4	4	B, M, V, W, D, N, R, G, J, Z
5	3	C, X, G, J, Z, S
6	5	D, N, R, J, Z, C, S, L, T
7	-	E, I, Y
8	2	Není samohláska
9	-	-ING, -ER, -ES, -ED, -ELY, -EFUL

V případě, že je proměnná ZnakTab rovna hodnotě 6, tak je ověřeno, jestli je ASCII znak jedním z ASCII znaků D, N, R, J, Z, S, L, T. Pokud jím je, tak je návratová hodnota funkce 1. V případě, kdy není jedním z vyjmenovaných ASCII znaků, ale je ASCII znakem H, tak je opět odečtena nebo přičtena 1 k proměnné Poloha podle nastaveného režimu ověřování.

A pokud je ASCII znak na nové poloze v globálním poli TextUpraveny ASCII znakem C, S nebo T, tak je návratová hodnota funkce 1.

Když je proměnná ZnakTab rovna hodnotě 7, nedochází k ověření žádného bitu, ale je rovnou porovnávána proměnná Znak s ASCII znaky E, I a Y. Pokud je v proměnné Znak uložen jeden z těchto ASCII znaků, tak je návratová hodnota funkce 1.

V případě, že je proměnná ZnakTab rovna 8, tak jsou postupně ověřeny všechny předchozí nebo následující ASCII znaky v globálním poli TextUpraveny dokud daný ASCII znak není souhláskou. Vždy je návratová hodnota funkce 1.

V případě splnění podmínek vrací funkce hodnotu 1 a v případě nesplnění podmínek vrací funkce hodnotu 0.

Funkce FonemTransNerovno

Tato funkce má jednu vstupní proměnnou Poloha a je v ní procházena tabulka FoneticTrans od aktuálního prvku části slova, který je určen proměnou Poloha až po hodnotu 0, která značí konec aktuální části slova. Potom je zvětšena hodnota proměnné Poloha o 1 a tato hodnota je vrácena jako návratová hodnota zpět do funkce FonemTransAlg.

Funkce FonemTransRovno

Tato funkce má 2 vstupní proměnné, kterými jsou reference na proměnou PolohaTab a reference na proměnou PolohaFonem. Reference na proměnou PolohaTab značí aktuální polohu v tabulce FoneticTrans a reference na proměnnou PolohaFonem ukazuje na polohu v globálním poli FoneticZapis. Obě proměnné jsou postupně zvětšovány o 1 a do globálního pole FoneticZapis je při tom postupně vkládán fonetický zápis aktuální části slova z tabulky FoneticTrans.

4.1.4 Úprava fonetického zápisu

V komponentě úprava fonetického zápisu jsou přidány fonémy za dvojhlásky, ploziva a afrikáty. Dále jsou například upraveny hodnoty prodloužení, zkrácení a tónu u souhlásek, které leží před samohláskou. V případě, že je nastaven režim 0, je vstupním fonetickým zápisem globální pole FoneticZapis.

A pokud je nastaven režim 1, je vstupním fonetickým zápisem globální proměnná Text. Komponenta Úprava fonetického zápisu je pro režimy 2 a 3 vynechána.

Úpravu fonetického zápisu tvoří funkce FonemyUprava, ve které je postupně voláno 6 funkcí, které jsou popsány v následujících podkapitolách.

Funkce FonemyPole

Ve funkci FonemyPole je převeden fonetický zápis z formátu, ve kterém jsou jednotlivé fonémy a jejich možné dodatečné nastavení, jako jeho délka a tón, řazeny za sebou, do formátu, kde každý foném v poli má vlastní strukturu s informacemi.

Tato struktura obsahuje název fonému, jeho umístění v tabulce ParametryFonemy, jeho délku, tón, počet zvuků a nakonec informaci o tom zda je souhláskou nebo samohláskou. Umístění v tabulce ParametryFonemy tvoří 3 hodnoty. Z těchto struktur je vytvořeno globální pole Fonemy.

Algoritmus převodu formátů fonetického zápisu je proveden postupným procházením vstupního fonetického zápisu a hledáním aktuálního fonému v tabulce FonemySeznam. Jelikož velká část fonémů je tvořena 2 ASCII znaky, je 1. provedeno hledání 1. ASCII znaku fonému, a když je nalezena shoda, tak je ověřeno jestli následující ASCII znak ve vstupním fonetickém zápise a 2. ASCII znak fonému v tabulce FonemySeznam jsou shodné. Když jsou shodné, tak je uložen aktuální foném fonetického zápisu do pole Fonemy. Zároveň s názvem fonému je uložena informace o tom, jestli je souhláska nebo samohláska a jeho umístění v tabulce ParametryFonemy. Umístění v tabulce ParametryFonemy se shoduje s umístěním fonému v tabulce FonemySeznam. Pokud se 2. ASCII znaky neshodují, tak pokračuje hledání 1. ASCII znaku fonému v tabulce FonemySeznam.

U fonémů, které tvoří jenom 1 ASCII znak je ověřeno pouze, jestli je následující ASCII znak aktuálního fonému v tabulce FonemySeznam mezerou. Potom následuje posun na následující ASCII znak ve vstupním fonetickém zápise.

Pokud je následující ASCII znak ve vstupním fonetickém zápise ASCII znakem čísel 1 až 9, tak je převeden na číslo a uložen do tónu aktuálního fonému v poli Fonemy. A když je následující ASCII znak ASCII znakem >, tak je nastaveno prodloužení aktuálního fonému v poli Fonemy. V případě, že je následující ASCII znak ASCII znakem <, tak je nastaveno naopak zkrácení. Jinak následuje přechod na hledání dalšího fonému.

Jelikož může být vstupním fonetickým zápisem i přímo fonetický zápis od uživatele je součástí algoritmu i převod malých písmen na velké pro případ, že uživatel zadal fonetický zápis malými písmeny.

Na konci převodu je za poslední foném v globálním poli Fonemy vložen foném Q, který tvoří konce řeči. A do globální proměnné FonemyPocet je uložen počet fonémů.

Funkce FonemyUmisteni

Po funkci FonemyPole následuje funkce FonemyUmisteni, ve které jsou přidány ukončovací fonémy za dvojhlásky nebo změněny některé fonémy v závislosti na tom, kde ve slově leží.

Ve funkci je postupně procházeno globální pole Fonemy a pokud aktuální foném splňuje jednu z podmínek vypsaných v tabulce 4.3, tak dojde buď k jeho úpravě nebo k vložení dalšího fonému za aktuální foném pomocí funkce FonemVlozeni.

Funkce FonemVlozeni má 2 vstupní proměnné, kterými jsou pozice, na kterou má být vložen daný foném a umístění požadovaného fonému v tabulce FonemySeznam. Vkládaný a aktuální foném mají stejné parametry jenom se liší jejich název a umístění v tabulce ParametryFonemy.

Tab. 4.3: Úpravy fonémů ve funkci FonemyUmisteni

Fonémy	Popis
EY, AY, OY,	Vložení ukončovacího fonému YX za aktuální foném
AW, OW, UW,	Vložení ukončovacího fonému WX za aktuální foném
UL	Změna fonému na foném AX a vložení fonému L
UM	Změna fonému na foném AX a vložení fonému M
UN	Změna fonému na foném AX a vložení fonému N
IL	Změna fonému na foném IX a vložení fonému L
IM	Změna fonému na foném IX a vložení fonému M
IN	Změna fonému na foném IX a vložení fonému N
R	Změna fonému na foném RX, pokud před ním ani za ním není samohláska
L	Změna fonému na foném LX, pokud před ním ani za ním není samohláska
S	Změna fonému na foném Z, pokud před ním je foném G
S	Pokud před fonémem S leží foném P, tak je změněn foném P na foném B, pokud před ním je samohláska
S	Pokud před fonémem S leží foném T, tak je změněn foném P na foném D, pokud před ním je samohláska
S	Pokud před fonémem S leží foném K, tak je změněn foném K na foném G, pokud před ním je samohláska

Funkce FonemySouhlaska

Ve funkci FonemySouhlaska je postupně procházeno globální pole Fonemy a ověřováno zda je aktuální foném souhláskou, ale přitom není souhláskou S. Pokud tyto podmínky splňuje a následující foném je samohláska, tak je jeho prodloužení, zkrácení a tón nastaven na stejné hodnoty jako má samohláska. Podmínkou pro změnu těchto parametrů je to, že má samohláska nastavený, alespoň 1 ze 3 výše vypsaných parametrů.

Funkce FonemyPloziv

Ve funkci FonemyPloziv jsou za ploziva a afrikáty přidány ukončovací fonémy. Pokud je aktuální foném jedním z afrikátů, tak je za něj přidán stejný foném, jenom jeho 2. hodnota parametru umístění v tabulce ParametryFonemy je změněna z 0 na 1, v případě, že aktuální afrikát má 2. hodnotu umístění rovnu 0. Když je 2. hodnota rovna 1, tak je změněna na číslo 2.

V případě, že je aktuální foném plozivem, tak je za něj vložen stejný foném jenom je jeho 2. hodnota parametru umístění v tabulce ParametryFonemy změněna z 0 na 1.

Funkce FonemyTon

Ve funkci FonemyTon je pro všechny fonémy nastaven počet zvuků, které ho budou tvořit a dále jsou upraveny hodnoty prodloužení, zkrácení a tónu. O tom kolik zvuků tvoří daný foném nerozhoduje pouze typ fonému, ale i jeho nastavený tón, prodloužení a zkrácení.

První je zkontrolováno jestli je nastaveno prodloužení a když nastaveno je, tak je do počtu zvuků aktuálního fonému vložena hodnota, která je vypočítaná sečtením 1. bajtu 1. hodnoty aktuálního fonému v tabulce ParametryFonemy s tou samou hodnotou jenom logicky posunutou o 1 doprava a číslem 1.

Když není nastaveno prodloužení, ale je nastaveno zkrácení, tak hodnotu počtu zvuků, tvoří součet 1. bajtu 1. hodnoty logicky posunuté o 1 s číslem 1. V případě, že není nastaveno prodloužení ani zkrácení, ale je nastaven tón fonému, tak počet zvuků tvoří 2. bajt 1. hodnoty aktuálního fonému v tabulce ParametryFonemy. Pokud není nastaven ani jeden ze 3 parametrů, tak je do počtu zvuků vložena hodnota 1. bajtu 1. hodnoty.

Ať už je nastavení prodloužení, zkrácení a tónu jakékoliv, vždy dojde k jejich úpravě. Úprava probíhá tak, že je k tónu přičteno prodloužení vynásobené číslem 128 a zkrácení vynásobené číslem 64. Dále je proveden logický součin výsledku sčítání s hodnotou 63 a nakonec je bit 7 výsledku vložen do proměnné prodloužení, bit 6 do proměnné zkrácení a první 4 bity jsou vloženy do proměnné tón.

Funkce FonemySpecial

Ve funkci FonemySpecial jsou upraveny počty zvuků u souhlásek, které leží před speciálními fonémy jako jsou například mezera, tečka a otazník. Změna počtu zvuků je provedena procházením globálního pole Fonemy, dokud není aktuální foném speciálním fonémem.

Když je aktuální foném speciálním fonémem, tak následuje procházení globálním polem Foném směrem zpět, dokud není nalezen foném, který není souhláskou.

Všechny fonémy, které jsou mezi samohláskou nebo dalším speciálním fonémem a aktuálním speciálním fonémem, mají upravený počet zvuků, pokud nejsou jedním ze souhlásek S, F, TH, SH nebo CH.

Úprava počtu zvuků je provedena přičtení počtu zvuků daného fonému logicky posunutého o 1 a čísla 1 k počtu zvuků daného fonému.

4.1.5 Tvorba řídicích parametrů

V komponentě tvorba řídicích parametrů jsou z globálního pole Fonemy vypočteny řídicí parametry pro výpočet amplitud znělých a neznělých zvuků generované řeči a pro nastavení generátoru šumu.

Pro jednotlivé prvky globálního pole Fonemy jsou v tabulce ParametryFonemy uloženy hodnoty, ze kterých jsou vypočteny řídicí parametry. Výsledné řídicí parametry ovlivňuje kromě hodnot v tabulce ParametryFonemy i nastavený tón, délka fonému a s jakými fonémy foném sousedí. Každý foném má v tabulce ParametryFonemy 16 hodnot, kromě ploziv, které mají 32 hodnot a afrikátů, které mají 48 hodnot.

Vypočtené řídicí parametry jsou uloženy do globálního pole Parametry a pro generování jednoho zvuku je potřeba 9 parametrů, jejichž funkce jsou vypsány v tabulce 4.4. Globální pole Parametry je dvojrozměrné pole, ve kterém 1. rozměr tvoří jednotlivé zvuky a 2. rozměr je tvořen 9 parametry pro 1 zvuk. Pole Parametrů je na začátku tvorby řídicích parametrů vynulováno.

Tab. 4.4: Konstrukce parametrů pro 1 zvuk generované řeči v komponentě tvorby řídicích parametrů

Parametr	Funkce
1.	Změna výpočtu umístění v tabulce AmplitudaUmisteni u znělého zvuku
2.	Amplituda kanálu C u neznělého zvuku
3.	Perioda generátoru šumu u neznělého zvuku
4. - 9.	Parametry pro výpočet umístění v tabulce AmplitudaUmisteni

Komponenta tvorby řídicích parametrů je složena ze 3 hlavních částí. V 1. části jsou vypočítány parametry pro výpočet umístění v tabulce AmplitudaUmisteni a amplituda 3. kanálu. Následně jsou nastaveny hodnoty periody generátoru šumu a nakonec vypočítán parametr změny výpočtu umístění v tabulce AmplitudaUmisteni u znělých zvuků.

První částí tvorby řídicích parametrů je cyklus, který je opakován pro každý prvek globálního pole Fonemy. Tento cyklus je složen z funkcí ParamVlozeni, ParamVypocet1, ParamVypocet2, ParamUprava a ParamHodZmena, které jsou postupně volány.

K uložení hodnot pro výpočet aktuálních řídicích parametrů je použito pole Promenne, které má 14 hodnot a jehož 11. pozice je před začátkem cyklu vynulována. Po výpočtu parametrů pro 1 foném je k proměnné UmisteniUprava přičten počet zvuků aktuálního fonému. Proměnná UmisteniUprava určuje, kde v globálním poli Parametry začíná zápis parametrů nového fonému. Po ukončení cyklu je proměnná UmisteniUprava uložena do proměnné ParamPocet, která určuje celkový počet zvuků.

Druhou část tvorby řídicích parametrů tvoří funkce ParamNoise a poslední část tvoří funkce ParamZmena. Jednotlivé funkce jsou popsány níže. Vývojový diagram tvorby řídicích parametrů je zobrazen na obrázku C.2.

Funkce ParamVlozeni

Ve funkci ParamVlozeni jsou vloženy vybrané hodnoty aktuálního fonému z tabulky ParametryFonemy do pole Promenne. Podrobný popis vybraných hodnot aktuálního fonému a jejich pozice v poli Promenne je v tabulce 4.5.

Tab. 4.5: Tabulka vybraných hodnot aktuálního fonému a jejich umístění v poli Promenne ve funkci ParamVlozeni

Umístění	Vložená hodnota
2. pozice	2. hodnota aktuálního fonému
1. bajt 3. pozice	1. bajt 10. hodnoty aktuálního fonému
5. pozice	4. hodnota aktuálního fonému
6. pozice	1. bajt 11. hodnoty aktuálního fonému
8. pozice	6. hodnota aktuálního fonému
9. pozice	1. bajt 12. hodnoty aktuálního fonému
10. pozice	1. bajt 13. hodnoty aktuálního fonému
13. pozice	vynulována
14. pozice	vynulována

Po vložení vybraných hodnot aktuálního fonému do pole Promenne následuje porovnání počtu zvuků aktuálního fonému s 2. bajtem 11. pozice v poli Promenne.

Když je počet zvuků menší než 2. bajt 11. pozice, tak je vložen na danou pozici v poli Promenne. Následně je porovnán počet zvuků aktuálního fonému s 1. bajtem 11. pozic a pokud je menší, tak je vložen na danou pozici v poli Promenne.

Funkce ParamVypocet1

Po funkci ParamVlozeni následuje funkce ParamVypocet1, ve které jsou vypočítány hodnoty 4., 6. a 8. parametru zvuku. Funkce ParamVypocet1 začíná výpočtem umístění začátku zápisu řídicích parametrů do globálního pole Parametry.

Umístění tvoří součet hodnoty 2. bajtu 13. pozice v poli Promenne a proměnné UmisteniUprava. Po výpočtu umístění je nastaven počet zvuků, do kterých budou vkládány nové řídicí parametry. Počet zvuků určuje 2. bajt 11. pozice v poli Promenne.

Po nastavení počtu zvuků je pro každý ze 3 řídicích parametrů volána funkce ParamVypocet1Alg, která má 5 vstupních proměnných. Název vstupních proměnných a vložené hodnoty pro jednotlivé parametry jsou vypsány v tabulce 4.6.

Tab. 4.6: Tabulka hodnot vstupních proměnných funkce ParamVypocet1Alg

Proměnná	4. parametr	6. parametr	8. parametr
1. proměnná	1. pozice	4. pozice	7. pozice
2. proměnná	2. pozice	5. pozice	8. pozice
3. proměnná	Počet zvuků	Počet zvuků	Počet zvuků
Poloha	Umístění začátku	Umístění začátku	Umístění začátku
Parametr	4	6	8

Ve funkci ParamVypocet1Alg je jako první ověřeno, že 3. proměnná není rovna 0. V případě, že je rovna 0, tak zápis daného řídicího parametru pro daný zvuku nenastane. Následně je do proměnné MezivysledekDel vložen výsledek dělení rozdílu 2. proměnné a 1. proměnné hodnotou 3. proměnné. Potom je výsledek dělení aritmeticky posunut o 1 doprava a je k němu přičtena 1. proměnná. Výsledek sčítání je vložen do proměnné Mezivysledek.

Po výpočtu proměnných MezivysledekDel a Mezivysledek následuje výpočet a zápis hodnot do daných řídicích parametrů zvuků, který je tvořen cyklem opakujícím se dokud není 3. parametr menší než 0.

V každém běhu cyklu je sečtena hodnota proměnné Mezivysledek s číslem 16. Výsledek sčítání je logicky posunut o 5 bitů doprava a vložen do proměnné Hodnota.

Pokud je daný řídicí parametr aktuálního zvuku roven 0, tak je do něj vložen 1. bajt proměnné Hodnota a následně je upravena proměnná Mezivysledek přičtením proměnné MezivysledekDel.

V případě, že daný řídicí parametr není roven 0, tak je ověřeno zda je proměnná MezivysledekDel větší nebo rovna 0. Když je větší nebo rovna 0, tak je daný řídicí parametr porovnáván s 1. bajtem proměnné Hodnota.

V případě, že je 1. bajt proměnné Hodnota větší, tak následuje uložení 1. bajtu proměnné Hodnota do daného řídicího parametru aktuálního zvuku. V obou případech následuje přičtení proměnné MezivysledekDel k proměnné Mezivysledek.

Když je proměnná MezivysledekDel menší než 0, tak následuje stejný postup, pouze k uložení 1. bajtu proměnné Hodnota do daného řídicího parametru aktuálního zvuku nastane v případě, že je 1. bajt proměnné Hodnota menší než hodnota daného parametru aktuálního zvuku. Po úpravě proměnné Mezivysledek následuje výpočet a úprava daného parametru následujícího zvuku. Vývojový diagram funkce ParamVypocet1Alg je zobrazen na obrázku C.3.

Funkce ParamVypocet2

Ve funkci ParamVypocet2 je nastaven 2., 5., 7. a 9. řídicí parametr. Funkce má stejný průběh jako funkce ParamVypocet1 pouze hodnota proměnné umístění je vypočítána přičtením 2. bajtu 14. pozice pole Promenne k proměnné UmisteniUprava a počet zvuků je určen hodnotou 1. bajtu 11. pozice pole Promenne.

Místo funkce ParamVypocet1Alg je pro 4 řídicí parametry volána funkce ParamVypocet2Alg, jejíž názvy stupních proměnných a vložené hodnoty pro jednotlivé řídicí parametry jsou vypsány v tabulce 4.7.

Tab. 4.7: Tabulka hodnot vstupních proměnných funkce ParamVypocet2Alg

Proměnná	2. parametr	5. parametr	7. parametr	9. parametr
1. proměnná	2. bajt 10. pozice	2. bajt 3. pozice	2. bajt 6. pozice	2. bajt 9. pozice
2. proměnná	1. bajt 10. pozice	1. bajt 3. pozice	1. bajt 6. pozice	1. bajt 9. pozice
3. proměnná	Počet zvuků	Počet zvuků	Počet zvuků	Počet zvuků
Poloha	Umístění začátku	Umístění začátku	Umístění začátku	Umístění začátku
Parametr	2	5	7	9

Funkce ParamVypocet2Alg se od funkce ParamVypocet1Alg liší pouze ve způsobu výpočtů proměnných MezivysledekDel, Mezivýsledek, Hodnota a úpravě proměnné Mezivýsledek před přechodem na zápis daného parametru do následujícího zvuku.

Jako první je vypočítána proměnná Rozdil, která je vypočítána odečtením 1. proměnné od 2. proměnné a logickým posunem výsledku odčítaní o 8 bitů doleva. Proměnná MezivysledekDel je následně vypočítána vydělením proměnné Rozdil 3. proměnnou a aritmetickým posunem výsledku dělení o 8 bitů doleva.

Proměnná Mezivýsledek je potom vypočítána aritmetickým posunem proměnné MezivysledekDel o 1 bit doprava, následným přičtením 1. proměnné, která má přehozené 1. a 2. slovo a nakonec prohozením 1. a 2. slova výsledku sčítání.

Proměnná Hodnota je vypočítána vynásobením 1. bajtu proměnné Mezivýsledek číslem 3 a odečtením čísla 89 od výsledku násobení.

Proměnná Mezivysledek je před přechodem na další zvuk upravena prohozením jejího 1. a 2. slova, následným přičtením proměnné MezivysledekDel a opětovným prohozením jejího 1. a 2. slova.

Funkce ParamNasledujici

Po funkci ParamVypocet2 následuje funkce ParamNasledujici, ve které jsou upraveny hodnoty v poli Promenne podle následujícího fonému. Tato funkce je rozdělena na 2 části. První část je provedena pro všechny fonémy v poli Fonemy a 2. část je provedena pro všechny fonémy kromě posledního fonému v poli Fonemy.

V 1. části jsou změněny pozice vybraných hodnot v poli Promenne podle tabulky 4.8.

Tab. 4.8: Změna umístění hodnot v poli Promenne ve funkci ParamNasledujici

Původní pozice	Nová pozice
2. bajt 11. pozice	2. bajt 13. pozice
1. bajt 11. pozice	2. bajt 14. pozice
2. pozice	1. pozice
5. pozice	4. pozice
8. pozice	7. pozici
1. bajt 3. pozice	2. bajt 3. pozice
1. bajt 6. pozice	2. bajt 6. pozice
1. bajt 9. pozice	2. bajt 9. pozice
1. bajt 10. pozice	2. bajt 10. pozice

V 2. části jsou změněny hodnoty v poli Promenne na základě hodnot aktuálního nebo následujícího fonému v tabulce ParametryFonemy. Druhá část začíná porovnáním 2. bajtů 8. hodnot aktuálního a následujícího fonému v tabulce ParametryFonemy.

Pokud je 2. bajt 8. hodnoty aktuálního fonému menší než 2. bajt 8. hodnoty následujícího fonému, tak dojde k úpravě hodnot v poli Promenne podle 1. způsobu v tabulce 4.9 a k následnému prohození umístění aktuálního a následujícího fonému v tabulce ParametryFonemy.

V případě, že 2. bajt 8. hodnoty aktuálního fonému není menší než 2. bajt 8. hodnoty následujícího fonému dojde k úpravě hodnot podle 2. způsobu v tabulce 4.9 a k prohození umístění nedochází.

Tab. 4.9: Vložení hodnot do pole Promenne podle 2. bajtu 8. hodnoty aktuálního a následujícího fonému ve funkci ParamNasledujici

Umístění	Vložená hodnota
1. způsob	
2. bajt 12. pozice	2. bajt 10. hodnoty následujícího fonému
2. bajt 11. pozice	1. bajt 9. hodnoty následujícího fonému
1. bajt 12. pozice	1. bajt 15. hodnoty následujícího fonému
1. bajt 11. pozice	2. bajt 15. hodnoty následujícího fonému
2. způsob	
2. bajt 11. pozice	2. bajt 10. hodnoty aktuálního fonému
2. bajt 12. pozice	1. bajt 9. hodnoty aktuálního fonému
1. bajt 11. pozice	1. bajt 15. hodnoty aktuálního fonému
1. bajt 12. pozice	2. bajt 15. hodnoty aktuálního fonému

Potom následuje úprava hodnoty 11. pozice pole Promenne. V případě, že je 1. bajt 11. pozice větší než nastavený počet zvuků aktuálního fonému, tak je nahrazen daným počtem zvuků. Stejný postup je opakován pro úpravu 2. bajtu 11. pozice.

Následující část funkce ParamNasledujici je prováděna až sedmkrát a liší se pouze v tom, které hodnoty jsou nastaveny jako vstupní proměnné funkce ParamNasledujiciAlg a která hodnota je přičtena k výsledné hodnotě a na kterou pozici v poli Promenne je výsledek vložen.

Funkce ParamNasledujiciAlg má 2 vstupní proměnné. Druhá proměnná určuje jak bude upravena 1. proměnná. V případě, že je 2. proměnná rovna 0, tak je do 1. proměnné vložena 0. Když 2. proměnná není rovna 0, tak je zmenšena o 1 a opět následuje ověření, zda je rovna 0. V případě, že je rovna 0, tak následuje aritmetický posun 1. proměnné o 1 doprava. Jinak nedochází k žádné úpravě 1. proměnné.

K 1. proměnné je následně přičtena vybraná hodnota z pole Promenne a výsledek je vložen na vybranou pozici v poli Promenne.

Všechny informace o tom, které hodnoty jsou vkládány do vstupních proměnných funkce ParamNasledujiciAlg, jaké hodnoty jsou přičítány a na kterou pozici jsou výsledky vkládány jsou v tabulce 4.10. První řádek tabulky je proveden jenom v případě, že následující foném není foném R. V případě, že je následující foném fonémem R, tak dojde pouze k vložení přičítané hodnoty na vybranou pozici v poli Promenne.

Tab. 4.10: Informace pro funkci ParamNasledujiciAlg a vložení hodnot do pole Promenne ve funkci ParamNasledujici

	1. proměnná (následující foném)	2. proměnná (aktuální foném)	Hodnota aktuálního fonému	Pozice v poli Promenne
1.	2. hodnota	1. bajt 8. hodnoty	3. hodnota	2. pozice
2.	4. hodnota	1. bajt 8. hodnoty	5. hodnota	5. pozice
3.	6. hodnota	2. bajt 9. hodnoty	7. hodnota	8. pozice
4.	1. bajt 10. hodnoty	1. bajt 14. hodnoty	2. bajt 11. hodnoty	1.bajt 3. pozice
5.	1. bajt 11. hodnoty	1. bajt 14. hodnoty	2. bajt 12. hodnoty	1.bajt 6. pozice
6.	1. bajt 12. hodnoty	1. bajt 14. hodnoty	2. bajt 13. hodnoty	1.bajt 9. pozice
7.	1. bajt 13. hodnoty	1. bajt 14. hodnoty	2. bajt 14. hodnoty	1.bajt 10. pozice

Funkce ParamUprava

Ve funkci ParamUprava je upraven 2., 4., 5., 6., 7., 8. a 9. řídicí parametr. Jako první je vypočítáno umístění v globálním poli Parametry, které je určené součtem 2. bajtu 13. pozice v poli Promenne s proměnou UmisteniUprava.

Úprava vybraných řídicích parametrů je rozdělena na 2 části. V 1. části je upraven 4., 6. a 8. parametr a k jejich úpravě dojde pouze, když je rozdíl počtu zvuků aktuálního fonému s 2. bajtem 13. pozice a s 2. bajtem 11. pozice větší než 0.

Výsledek odčítání určuje i počet zvuků, ve kterých mají být upraveny dané řídicí parametry.

Úprava 4., 6. a 8. řídicího parametru zvuku začíná přičtením čísla 16 k hodnotě na 1. pozici v poli Promenne a následným logickým posunutím výsledku sčítání o 5 bitů doprava. První bajt výsledku je následně vložen do 4. řídicího parametru.

Stejným způsobem je upravena hodnota 4. pozice v poli Promenne pro úpravu 6. řídicího parametru a hodnota 7. pozice pro úpravu 8. řídicího parametru.

Před přechodem na úpravu následujícího zvuku je k 2. bajtu 13. pozice v poli Promenne přičteno číslo 1.

V 2. části je upraven 2., 5., 7. a 9. řídicí parametr a k jejich úpravě dojde pouze tehdy, když je rozdíl počtu zvuků aktuálního fonému s 2. bajtem 14. pozice a s 1. bajtem 11. pozice větší než 0. Výsledek odčítání opět určuje v kolika zvucích jsou upraveny řídicí parametry.

Úprava 2., 5., 7. a 9. řídicího parametru zvuku začíná vynásobením 2. bajtu 9. pozice v poli Promenne číslem 3 a následným odečtením čísla 89. Pokud je výsledek menší nebo roven 0, tak je do 2. řídicího parametru vložena 0. Jinak je výsledek odčítání logicky posunut o 2 bity doprava a vložen do 2. řídicího parametru.

Pro 5. řídicí parametr je upravena hodnota 2. bajtu 3. pozice, pro 7. řídicí parametr je upraven 2. bajt 6. pozice a pro 9. řídicí parametr je upraven 2. bajt 9. pozice. Před přechodem na úpravu řídicích parametrů následujícího zvuku je k 2. bajtu 14. pozice přičteno číslo 1.

Po funkci ParamUprava opět následují funkce ParamVypocet1 a ParamVypocet2.

Funkce ParamHodZmena

Jako poslední je před přechodem na následující foném provedena úprava hodnot v poli Promenne, která je provedena přesunem hodnot mezi jednotlivými pozicemi. Přehled přesunutých hodnot je v tabulce 4.11.

Tab. 4.11: Změna umístění hodnot v poli Promenne ve funkci ParamHodZmena

Původní pozice	Nová pozice
2. pozice	1. pozice
5. pozice	4. pozice
8. pozice	7. pozice
1. bajt 3. pozice	2. bajt. 3. pozice
1. bajt 6. pozice	2. bajt. 6. pozice
1. bajt 9. pozice	2. bajt 9. pozice
1. bajt 10. pozice	2. bajt 10. pozice
12. pozice	11. pozice

Funkce ParamNoise

Ve funkci ParamNoise jsou do 3. řídicích parametrů zvuků vloženy hodnoty 1. bajtů 16. hodnot fonémů z tabulky ParametryFonemy.

O tom do kolika zvuků je vložen 1. bajt 16. hodnoty daného fonému rozhoduje opět počet zvuků daného fonému a proměnná Umisteni, která je zvětšena o 1 po každém zápisu hodnoty do 3. řídicího parametru zvuku.

Funkce ParamZmena

Jako poslední jsou vypočítány hodnoty 1. řídicích parametrů pro znělé zvuky. Výpočet 1. řídicího parametru probíhá pro každý foném a to do jakých zvuků má být vložena vypočítaná hodnota 1. řídicího parametru určuje opět nastavený počet zvuků daného fonému. Pro výpočet hodnoty 1. řídicího parametru zvuku je použita proměnná Zmena, jejíž výchozí hodnota je 66.

Jako první je ověřeno, zda je aktuální foném mezerou. Pokud jí je, tak následuje rovnou zápis hodnot do 1. řídicího parametru.

Pro daný počet zvuků je do 1. řídicího parametru vložena proměnná Zmena, která je po vložení do aktuálního zvuku buď zmenšena o 1 v případě, že je větší než 66 a nebo zvětšena o 1, když je menší než 66.

Když aktuální foném není mezerou, ale je speciálním fonémem, tak následuje úprava 1. řídicího parametru předchozích zvuků. Tato úprava probíhá ve 2 fázích. V 1. fázi jsou přičítány k proměnné SpecialPosun, jejíž výchozí hodnota je -20, hodnoty počtu zvuků předchozích fonémů.

Takto je procházeno polem Fonemy dokud není aktuální foném dalším speciálním fonémem a nebo dokud není proměnná SpecialPosun kladná nebo rovna 0.

V případě, že bylo procházení polem Fonemy ukončeno z důvodu toho, že je proměnná SpecialPosun kladná nebo rovna 0, tak je proměnná SpecialPosun nastavena do 0.

Následně je k proměnné SpecialPosun přičteno číslo 20 a pokud je výsledek sčítání roven 0, tak je do proměnné Zmena vložena hodnota 66 a následuje stejný zápis hodnot do 1. řídicího parametru jako v případě, že aktuální foném je mezerou. Proměnná SpecialPosun určuje kolik předchozích zvuků má být upraveno.

V případě, že speciální foném nebyl tečkou ani otazníkem, tak následuje dodatečná úprava proměnné SpecialPosun. Při této úpravě je proměnná SpecialPosun logicky posunuta o 1 doprava a následně je k ní přičteno číslo 1.

Potom následuje samotná úprava 1. řídicího parametru předchozích zvuků, jejichž počet je dán proměnou SpecialPosun a která začíná úpravou nejvzdálenějšího zvuku. K hodnotě 1. řídicího parametru nejvzdálenějšího zvuku je přičteno číslo 1, které je pro úpravu každého následujícího zvuku zvětšeno o 1. V případě, že speciální foném byl otazníkem, není přičítána 1, ale -1, která je před zápisem každého dalšího zvuku zmenšena o 1.

Po úpravě předchozích zvuků je do proměnné Zmena vložena hodnota 66 a následuje opět zápis hodnot jak v případě, že je aktuální foném mezerou.

V případě, že aktuální zvuk není mezerou ani speciálním fonémem, ale má nastaven alespoň 1 z parametrů délky nebo tón, tak následuje 2. způsob zápisu hodnot do 1. řídicího parametru.

Tento zápis začíná sloučením tónu, prodloužení a zkrácení aktuálního fonému do 1 bajtu. Tón tvoří první 4 bity bajtu, prodloužení tvoří bit 7 a zkrácení tvoří bit 6. K takto vytvořenému bajtu je dvakrát přičten ten samý bajt a výsledek je vynásoben -1 a je k němu následně přičteno číslo 66.

Výsledek sčítání je potom vložen do všech 1. řídicích parametrů daných zvuků a do proměnné Zmena. V případě, že není foném mezerou, speciálním fonémem a ani nemá nastavenou délku nebo tón, tak jeho zvuky mají 1. řídicí parametr nastaven stejným způsobem jako by byl foném mezerou.

4.1.6 Generátor řeči

Komponentu Generátor řeči tvoří funkce ZvukGen, ve které jsou postupně generovány jednotlivé zvuky z pole Parametry. O tom, jestli je generován znělý nebo neznělý zvuk rozhoduje hodnota 2. řídicího parametru. Pokud je 2. řídicí parametr roven 0, tak jde o znělý zvuk, jinak jde o zvuk neznělý.

Jako první je vypočítána perioda generované řeči, která je získána vydělením hodnoty proměnné Intonace číslem 614400.

Po vypočítání periody generované řeči následuje cyklus, který je opakován pro každý zvuk v globálním poli Parametry. Jako první je vložena hodnota globální proměnné Tempo do proměnné ZvukZmena, která určuje kolikrát budou vypočítány amplitudy pro každý zvuk. Když je aktuální zvuk znělý, je proměnná ZvukZmena zvětšena o 1 a v případě, že je aktuální zvuk neznělý, tak je nastavena nová perioda generátoru šumu. Hodnota periody generátoru šumu je vložena do globální proměnné GenPeriod.

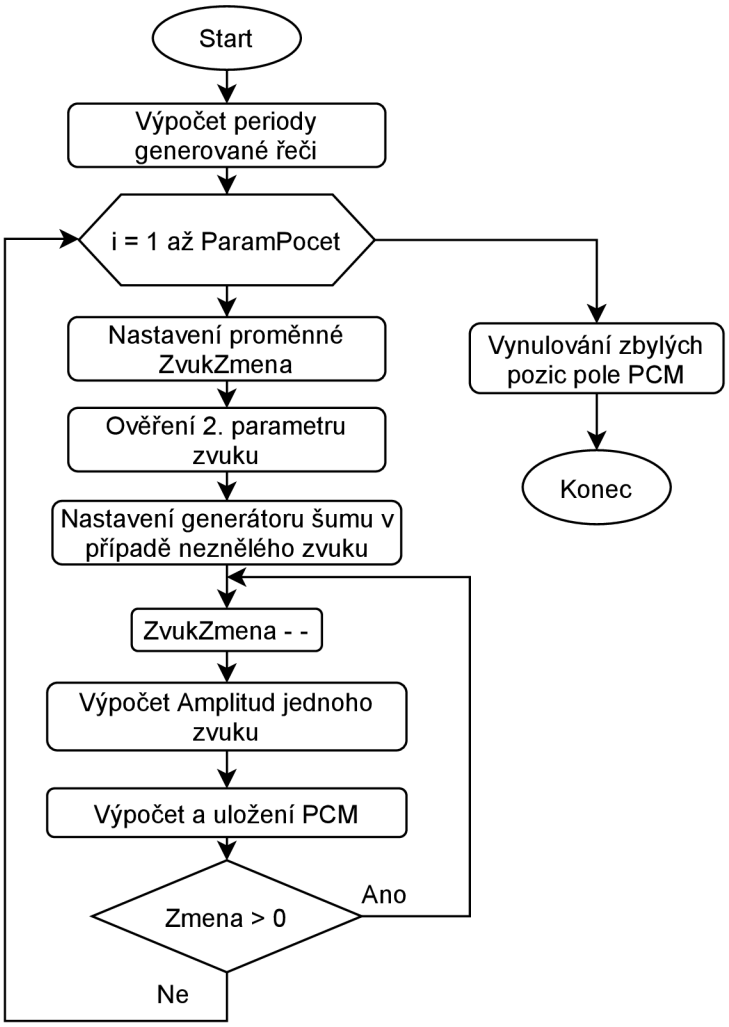
Po změně proměnné ZvukZmena a nastavení periody generátoru šumu, v případě neznělého zvuku, následuje cyklus, který je prováděn, dokud není proměnná ZvukZmena rovna 0. V 1 běhu cyklu je nejprve zmenšena hodnota ZvukZmena o 1 a v případě neznělého zvuku je volána funkce ZvukSumGen, která tvoří generátor šumu.

Po funkci ZvukSumGen následuje funkce ZvukNeznely, ve které jsou vypočítány hodnoty 3 amplitud pro výpočet výsledné úrovně pulzně kódové modulace u neznělého zvuku. Pokud je generován znělý zvuk, tak je volána pouze funkce ZvukZnely.

Výpočet výsledné úrovně pulzně kódové modulace je v obou případech stejný. První jsou jednotlivé amplitudy, které jsou uloženy v poli Amplitudy, převedeny z rozsahu 0 až 15 na rozsah 0 až 1 V pomocí funkce ZvukNapeti. V této funkci je nejprve vypočítán rozdíl čísla 15 a hodnoty dané amplitudy, který je potom vydělen číslem -2. Odpovídající hodnota napětí je následně vypočítána umocněním čísla 2 výsledkem dělení.

Výsledné hodnoty napětí jsou potom sečteny a vynásobeny číslem 255. Výsledek násobení je dále vydělen číslem 1,6, čímž je vytvořena odpovídající úroveň 8bitové pulzně kódové modulace. V případě neznělých zvuků rozhoduje o přičtení amplitudy 3. kanálu hodnota 3. prvku globálního pole GenSum generátoru šumu.

Výsledné hodnoty pulzně kódové modulace jsou ukládány do pole PCM. Po ukončení generovaní jsou zbylé pozice v poli PCM vynulovány, aby nedošlo k překrytí předchozí delší generované řeči s novou kratší. Vývojový diagram funkce ZvukGen je zobrazen na obrázku 4.3.



Obr. 4.3: Vývojový diagram funkce ZvukGen

Funkce ZvukSumGen

Ve funkci ZvukSumGen je hlavní algoritmus generátoru šumu. Generátor šumu je 17bitový zpětnovazební čítač, jehož zpětnou vazbu tvoří exkluzivní logický součet bitů 2 a 0. Jednotlivé bity 17bitové zpětnovazebního čítače tvoří pole GenSum.

Frekvence čítaní je dána globální proměnnou SumEn a hodnotou globální proměnné GenPeriod, která je vynásobená číslem 2.

Pokud je proměnná SumEn rovna nebo větší než globální proměnná GenPeriod, tak jsou posunuty hodnoty v poli GenSum o 1 doleva a na 16. pozici je vložena zpětná vazba. Nakonec je hodnota globální proměnné SumEn vynulována.

Funkce ZvukZnely

Ve funkci ZvukZnely jsou vypočítány hodnoty 3 amplitud znělého zvuku. Pro výpočet znělého zvuku je použit 4. až 9. řídicí parametr zvuku. 4., 6. a 8. řídicí parametr je použit pro výpočet 1. bajtu umístění v tabulce AmplitudaUmisteni a 5., 7. a 9. řídicí parametr je vložen jako 2. bajt umístění v tabulce AmplitudaUmisteni.

První bajt umístění je tvořen součtem posledních hodnot 1. bajtu umístění v globálním poli ZnelyPozice s hodnotou 4., 6. nebo 8. řídicího parametru aktuálního zvuku. Výchozí hodnoty v globálním poli ZnelyPozice jsou 0.

Pomocí takto vytvořených hodnot umístění jsou nalezeny hodnoty v tabulce AmplitudaUmisteni, které jsou následně sečteny a jejich součet je logicky posunut o 3 bity doleva. Tato hodnota potom tvoří umístění začátku dané kombinace amplitud v tabulce Amplituda. První amplituda je přímo na pozici, kterou udává vypočítaná hodnota umístění. Amplituda 2. kanálu je na pozici, kterou udává součet vypočítané hodnoty umístění s číslem 3 a amplituda 3. kanálu je na pozici, kterou udává součet vypočítané hodnoty s číslem 5.

Dále je u generování znělých zvuků použit 1. řídicí parametr, který je při každém generování znělého zvuku zmenšen o 1, a když je roven 0, tak je globální pole ZnelyPozice před přičtením aktuálních hodnot 4., 6. a 8. řídicího parametru vynulováno. Nalezené hodnoty amplitud jsou uloženy do pole Amplitudy.

Funkce ZvukNeznely

Ve funkci ZvukNeznely jsou vypočítány hodnoty 3 amplitud neznělého zvuku. Výpočet amplitud probíhá stejně jako u znělých zvuků pouze nedochází k vynulování globálního pole NeznelyPozice, ve kterém jsou uloženy poslední hodnoty 1. bajtu umístění v tabulce AmplitudaUmisteni.

Dále nedochází k uložení amplitudy 3. kanálu dané kombinace amplitud v tabulce Amplituda. Místo toho je do pole Amplitudy vložena hodnota 2. řídicího parametru.

5 Dokončení implementace programu a realizace softwarové knihovny

V rámci dokončení programové implementace programu pro hlasový výstup byla vytvořena hlavní funkce GenMain, ve které jsou postupně volány jednotlivé základní komponenty, které byli implementovány a popsány v kapitole 4.

Komponenty jsou volány v pořadí, které je zobrazeno v blokovém schématu na obrázku 4.1. Po dokončení každé komponenty je ověřeno, zda nedošlo během jejího provádění k chybě. Když k ní došlo, tak je ukončeno provádění funkce GenMain.

Dokončená programová implementace pro hlasový výstup byla následně ověřena zápisem výstupních hodnot pulzně kódové modulace do souboru typu .wav.

5.1 Zápis výstupních hodnot PCM do souboru .wav

Zápis do souboru typu .wav je realizován funkcí s názvem ZapisAudio, jejíž jedinou vstupní proměnou je cesta k souboru, do kterého mají být data uložena. Pokud daný soubor neexistuje, tak je ve vybraném adresáři vytvořen nový soubor se stejným názvem. Pro vytvoření a zápis do souboru jsou použity metody třídy ofstream knihovny fstream.

Funkce ZapisAudio začíná vypsáním hlavičky souboru, v níž se hodnoty jednotlivých parametrů mohou lišit podle nastavené intonace řeči. Po vypsání hlavičky souboru následuje zápis výstupních hodnot pulzně kódové modulace, které jsou uloženy v poli PCM o velikosti 200 000 hodnot. Příklad hlavičky souboru pro intonaci řeči nastavenou na hodnotu 77 je zobrazen v tabulce 5.1.

Zápisem do souboru typu .wav bylo ověřeno, že generovaná řeč je na poslech téměř nerozlišitelná od syntetické řeči, kterou generuje program Atari 520ST Speech Synthesizer V2.0 v emulátoru počítače Atari 520ST. Dále byla ověřena funkcionalita nastavení režimu, intonace a tempa řeči.

Generované řeči je možné rozlišit při generování frikativní souhlásky S, při které vzniká výrazný šum u syntetické řeči generované programem v emulátoru Atari 520ST. Tento rozdíl může být způsoben odlišnou implementací generátoru šumu.

To jak jsou generované řeči podobné nelze přesně určit, protože syntetická řeč generovaná programem v emulátoru Atari 520ST je dodatečně upravena tak, aby mohla být uložena do souboru typu .wav společně s jinými zvuky, které jsou emulátorem generovány. Těmito zvuky jsou například pípnutí při stlačení klávesy na klávesnici.

Mezi úpravy syntetické řeči vytvořené emulátorem patří převod z 8bitové pulzně kódové modulace na 16bitovou pulzně kódovou modulaci a úprava syntetické řeči pro vzorkovací frekvenci 44100 Hz. Dále mohla být upravena pomocí dodatečných filtrů, které původní osobní počítač Atari 520ST neobsahoval.

V rámci analýzy generované řeči byly vytvořeny grafy průběhů generované řeči, na kterých je možné vidět změnu hodnot pulzně kódové modulace v čase. Grafy průběhů slova "Hello" generovaného implementovaným hlasovým generátorem a programem v emulátoru Atari 520ST jsou zobrazeny na obrázku D.1. Na grafech lze vidět, že obě generované řeči mají podobný průběh, i když mají jiný rozsah pulzně kódové modulace a vzorkovací frekvence.

Tab. 5.1: Hlavička souboru typu .wav pro nastavenou hodnotu 77 intonace řeči

Parametr	Hodnota
Identifikátor	"RIFF"
Velikost souboru	200 036 bajtů
Identifikátor	"WAVE"
Formátová část	"fmt "
Velikost formátové části	16 bajtů
Formát zvuku	1
Počet kanálů	1
Vzorkovací frekvence	7 979 Hz
Datový tok	7 979 bajtů
Zarovnání bloku	1 bajt
Bitová hloubka	8 bitů
Datová část	"data"
Velikost datové části	200 000 bajtů

5.2 Realizace softwarové knihovny pro vybraný embedded systém

Z kódu implemenotvaného hlasového generátoru byla vytvořena softwarová knihovna pro vybraný embedded systém rozdělením kódu na hlavičkový soubor TTS.h a implementační soubor TTS.cpp.

V hlavičkovém souboru jsou jednotlivé funkce hlasového generátoru deklarované jako metody třídy CTTS, díky čemuž je možné mít více hlasových generátorů v jednom programu současně.

Kromě metod jsou v hlavičkovém souboru deklarované i proměnné. Mezi deklarovanými proměnnými jsou i tabulky FoneticTrans, ParametryFonemy, AmplitudaUmisteni a Amplituda, které jsou kvůli svému rozsahu uloženy v samostatných souborech a tím pádem jsou deklarované jako externí proměnné.

Vytvořená knihovna se tedy skládá celkem z 5 souborů typu .cpp a 1 souboru typu .h. Jednotlivé metody hlasového generátoru jsou v třídě CTTS rozděleny na veřejné a soukromé podle toho, jestli k nim uživatel má mít přístup.

V rámci realizace softwarové knihovny byla vytvořena funkce ZapisPWM, která generuje řeč na vybrané piny GPIO vývojové desky Raspberry Pi Pico.

Ve funkci ZapisPWM jsou jednotlivé hodnoty pulzně kódové modulace generovány pomocí pulzně šířkové modulace, při které je požadovaná hodnota výstupního napětí tvořena přepínáním digitálního signálu mezi logickou 1 a 0. O přechodu mezi logickou 1 a 0 rozhoduje, to jestli je aktuální hodnota pulzně kódové modulace větší nebo menší než hodnota volně čítajícího čítače [23].

Pro generování pulzně šířkové modulace byly použity funkce knihovny hardware_pwm. Algoritmus funkce ZapisPWM tvoří cyklus, ve kterém jsou postupně generovány hodnoty z pole PCM.

6 Demonstrační aplikace

V následujícím textu je popsáno blokové uspořádání demonstrační aplikace pro ovládání hlasového generátoru pomocí PC a demonstrační aplikace pro prezentaci funkcionality vytvořeného hlasového generátoru.

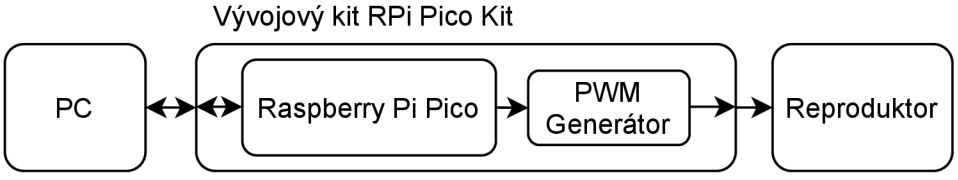
6.1 Návrh blokového uspořádání demonstrační aplikace

Navržené blokové uspořádání demonstrační aplikace pro ovládání hlasového generátoru pomocí PC tvoří vybraná vývojová deska Raspberry Pi Pico připojená k vývojovému kitu RPi Pico Kit. RPi Pico Kit je k PC připojen přes USB rozhraní vývojové desky Raspberry Pi Pico, které umožňuje nahrání programu a sériovou komunikaci UART. Jako audio výstup je použit PWM generátor zvuku vývojového kitu RPi Pico Kit, ke kterému je pomocí výstupu 3,5 mm jack připojen reproduktor[24]. Blokové uspořádání je zobrazeno na obrázku 6.1.

6.1.1 Vývojový kit RPi Pico Kit

RPi Pico Kit je vývojový kit, který byl vytvořen pro kurz Vestavné systémy a rozšiřuje vývojovou desku Raspberry Pi Pico o obvody, kterými jsou například tlačítka, zvukové obvody, VGA rozhraní a OLED displej. Součástí vývojového kitu je i mikrokontrolér Raspberry Pi RP20400, který umožňuje nahraní programu do vývojové desky pomocí SWD rozhraní a ladění nahraného programu[24].

Vývojový kit disponuje 3 zvukovými obvody, kterými jsou výše zmíněný PWM generátor zvuku, piezoelektrický měnič a PCM generátor zvuku. Výstupy generátorů zvuku jsou připojeny k výstupům 3,5 mm jack [24]. Podrobnější informace k vývojovému kitu RPi Pico Kit jsou v závěrečné práci [24].



Obr. 6.1: Blokové uspořádání demonstrační aplikace pro komunikaci s PC

Obr. 6.2: Fotografie zapojení vývojového kitu RPi Pico Kit s Raspberry Pi Pico pro komunikaci s PC

6.2 Realizace demonstračních aplikací

Pro demonstraci funkcionality realizovaného hlasového generátoru byly vytvořeny 2 demonstrační aplikace.

V 1. demonstrační aplikaci jsou výstupní hodnoty pulzně kódové modulace zapisovány do souboru typu .wav, který umožňuje prezentování funkcionality hlasového generátoru, i když uživatel nemá k dispozici potřebný hardware. Tato demonstrační aplikace je navržena tak, že uživatel může sám nastavit režim hlasového generátoru, vstupní text, intonaci i tempo generované řeči.

Ve funkci main 1. demonstrační aplikace je připravena 1 třída realizovaného hlasového generátoru. Pro danou třídu je demonstrováno volání funkce TempoIntonace pomocí, které může uživatel nastavit intonaci a tempo generované řeči.

Dále funkce main obsahuje 3 funkce GenMain pomocí, kterých je prováděn hlavní algoritmus hlasového generátoru. Po provedení funkce GenMain je vygenerovaná řeč uložena do souboru typu .wav pomocí funkce audioWrite. Uživatel může sám zvolit adresář a název souboru, do kterého má být vygenerovaná řeč uložena.

Funkce GenMain je volána v režimu generování řeči z anglického textu, následně je volána v režimu generování řeči z fonetického zápisu a nakonec je řeč generována z naposledy uložených dat. Po vygenerování řeči je na standardní výstup vypsán vstupní text, anglický text uložený v hlasovém generátoru a fonetický zápis aktuálně vygenerované řeči.

Druhá demonstrační aplikace prezentuje funkčnost implementovaného hlasového generátoru na blokovém uspořádání, které je zobrazeno na obrázku 6.1. Tuto demonstrační aplikaci tvoří program, který generuje syntetickou řeč z předem nastaveného anglického textu pomocí funkce ZapisPWM. Program je do vývojové desky Raspberry Pi Pico nahrán pomocí režimu velkokapacitního paměťového USB zařízení.

Pro obě demonstrační aplikace jsou v elektronické příloze k dispozici demonstrační nahrávky.

Závěr

Cílem práce bylo navrhnout a implementovat hlasový generátor, který bude tvořit softwarovou knihovnu pro zvolený embedded systém.

V 1. kapitole byla provedena rešerše existujících principů generování hlasového výstupu v počítačových systémech. V rámci rešerše byly popsány 3 základní metody generování hlasového výstupu v počítačových systémech. První popsanou metodou byla artikulační syntéza, která je nejobecnější metodou generování řeči a která se zabývá modelováním hlasového ústrojí člověka.

Jako 2. byla vysvětlena konkatenační syntéza, která funguje na principu řetězení řečových jednotek, což jsou malé segmenty nahrávek lidské řeči. Často používanými řečovými jednotkami jsou difony a demislabiky, které umožňují složení jakéhokoli slova. Difony jsou řečové jednotky, které začínají v polovině 1. fonému a končí v polovině fonému následujícího. A demislabiky jsou řečové jednotky, které začínají v polovině 1. slabiky a končí v polovině následující. Poslední popsanou základní metodou byla formantová syntéza, při které je řeč tvořena pomocí akusticko-fonetických pravidel, které generují řídicí parametry syntezátoru. Formantová syntéza má ze základních metod nejmenší paměťové nároky, které se pohybují maximálně v desítkách kB. Největší paměťové nároky má konkatenační syntéza, jejíž paměťové nároky se pohybují od desítek až po tisíce MB.

Kromě základních metod generování hlasového výstupu byla vysvětlena i syntéza řeči z textu. Její úlohou je zpracovat vstupní text a následně ho pomocí jedné ze základních metod syntézy řeči převést na syntetickou řeč. Syntéza řeči z textu se skládá ze 2 hlavních částí, kterými jsou analýza textu a syntéza řeči. Obě hlavní části jsou popsány v podkapitole 1.6.

V kapitole 2 byla analyzována architektura veřejně dostupného programu Atari 520ST Speech Synthesizer V2.0, který byl vytvořen v roce 1986. Program převádí psaný anglický text nebo fonetický zápis na syntetickou řeč. Dále je v programu možné nastavit tempo a intonaci generované řeči.

Atari 520ST je 16bitový osobní počítač od firmy Atari Corporation z roku 1985, obsahuje mikroprocesor architektury M68000 16/32bitových mikroprocesorů od firmy Motorola Inc., víceúčelovou periferii MC68901 a zvukový obvod AY-3-8910.

Program Atari 520ST Speech Synthesizer V2.0 musel být zpětně analyzován ze strojového kódu programu, protože již nejsou k dispozici zdrojové texty. Zpětná analýza strojového kódu programu byla provedena přeložením strojového kódu pomocí volně dostupného nástroje Ghidra [11]. Původní strojový kód měl velikost 28 kB a přeložený analyzovaný kód měl 195 kB ve formátu .gzf. Součástí zpětné analýzy programu bylo i analyzování programu v emulátoru počítače Atari 520ST.

Analýzou programu bylo zjištěno, že pro generování syntetické řeči byla pravděpodobně použita zjednodušená varianta konkatenační syntézy. Analýza programu byla časově náročná a zabrala celý zimní semestr a velkou část letního semestru.

Jako vhodný embedded systém pro realizaci knihovny hlasového generátoru byla vybrána vývojová deska Raspberry Pi Pico, která byla zvolena kvůli tomu, že je cenově dostupnou vývojovou deskou podporující programovací jazyk C/C++. Dále splňuje hardwarové požadavky pro generování audia. Mezi tyto požadavky patří například časovače s vysokou frekvencí časování nebo dostatečně velká paměť pro uložení programu a jeho dat(řádově stovky kB).

Po výběru vhodného embedded systému byl navržen hlasový generátor a implementovány jeho základní komponenty. Blokové schéma hlasového generátoru je zobrazeno na obrázku 4.1 na straně 55. Implementovaný hlasový generátor je složen ze 6 základních komponent, kde 5 komponent tvoří hlavní algoritmus hlasového generátoru. Intonace a tempo řeči jsou nastaveny samostatně, aby uživatel nemusel opakovaně zadávat jejich hodnoty i v případě, že nedochází k jejich změně.

U hlasového generátoru je možné nastavit 4 režimy, kterými jsou generování syntetické řeči z anglického textu, generování syntetické řeči z fonetického zápisu nebo z předchozích dat a poslední režim lze použít k testování ukončení generování syntetické řeči.

Dokončení implementace hlasového generátoru bylo realizováno v kapitole 5. Funkcionalita hlasového generátoru byla následně otestována zápisem výstupních hodnot hlasového generátoru do souboru typu .wav. Generovaná řeč je na poslech téměř nerozlišitelná od syntetické řeči, kterou generuje program v emulátoru počítače Atari 520ST a to i při změně nastavení intonace a tempa řeči.

Bohužel nebylo možné ověřit zda implementovaný hlasový generátor generuje řeč, která je zcela identická jako syntetická řeč původního programu, protože řeč generovaná programem je v emulátoru převedena z 8bitové pulzně kódové modulace na 16bitovou a je upravena pro vzorkovací frekvenci 44100 Hz. Dále mohla být upravena pomocí dodatečných filtrů, které původní osobní počítač Atari 520ST neobsahoval. V rámci analýzy byly vytvořeny grafy průběhů generované řeči, které jsou zobrazeny na obrázku D.1.

Implementovaný hlasový generátor je možné použít jako doplňkový způsob signalizace a nebo všude, kde nejsou ostatní druhy signalizace vhodné. Implementace hlasového generátoru zabrala velkou část letního semestru a zdrojové soubory realizovaného hlasového generátoru mají přes 120 kB.

Jako poslední bylo navrženo blokové uspořádání demonstrační aplikace pro ovládání hlasového generátoru pomocí PC a vytvořeny 2 demonstrační aplikace. Blokové uspořádání tvoří vývojová deska Raspberry Pi Pico, která je připojena k vývojovému kitu RPi Pico Kit [24]. Vývojový kit je k PC připojen pomocí USB rozhraní.

Blokové uspořádání demonstrační aplikace je zobrazeno na obrázku 6.1. První demonstrační aplikace byla realizována zápisem výstupu hlasového generátoru do souboru typu .wav, což umožňuje prezentování funkcionality hlasového generátoru, i když není k dispozici potřebný hardware. Druhá demonstrační aplikace byla realizována pro navržené blokové uspořádání demonstrační aplikace a také generuje řeč, která na poslech zní téměř nerozlišitelně od řeči generované programem Atari 520ST Speech Synthesizer V2.0. Pro obě demonstrační aplikace jsou v elektronické příloze k dispozici demonstrační nahrávky.

V pokračování práce je možné rozšířit implementovaný hlasový generátor o fonémy umožňující generování českého jazyka nebo rozšířit tabulku pro převod anglického textu na fonetický zápis o další zkratky, názvy společností a čísla větší než 9. Dále je možné upravit realizovanou softwarovou knihovnu pro další embbeded systémy.

Literatura

[1] JURAFSKY, Daniel a James H. MARTIN. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition* [online]. 3rd ed. draft. 2023, 636 s. [cit. 2023-02-15]. Dostupné z: `https://web.stanford.edu/~jurafsky/slp3/ed3book_jan72023.pdf`

[2] ČADA, Ondřej. *Mikroprocesor Motorola 68000: příručka programátora.* Praha: Grada, 1992. Nestůjte za dveřmi. ISBN 80-85424-64-9.

[3] BROWN, Dalvin. *AI gave Val Kilmer his voice back. But critics worry the technology could be misused.*The Washington Post [online]. 18 August 2021 [cit. 2022-12-10]. Dostupné z: `https://www.washingtonpost.com/technology/2021/08/18/val-kilmer-ai-voice-cloning/`

[4] *What is Speech Synthesis? A Detailed Guide.* WebsiteVoice [online]. 24 Aug 2022 [cit. 2022-12-10]. Dostupné z: `https://websitevoice.com/blog/what-is-speech-synthesis/`

[5] HOLMES, John a Wendy HOLMES. *Speech synthesis and recognition* .2nd ed. London ; New York: Taylor & Francis, 2001, 298 s. ISBN 0-7484-0856-8.

[6] RODMAN, Robert D. *Computer Speech Technology.* Boston: Artech House, 1999, 344 s. : il. ISBN 0-89006-297-8.

[7] PSUTKA, Josef et al. *Mluvíme s počítačem česky.* Praha: Academia, 2006, 746 s. ISBN 80-200-1309-1.

[8] *The voder: The first electronic voice synthesizer* [online]. SASO, 2022 [cit. 2022-10-23]. Dostupné z: `https://www.whatisthevoder.com`

[9] TATHAM, Mark a Katherine MORTON. *Developments in speech synthesis.* Chichester: John Wiley, 2005, 342 s. ISBN 0-470-85538-X.

[10] BEVERIDGE, A.D. a M.N. DAY. *Atari 520ST Speech Synthesizer V2.0. Demozoo* [online]. [cit. 2022-12-16]. Dostupné z: `https://demozoo.org/productions/125048/`

[11] NATIONAL SECURITY AGENCY. *Ghidra* [online]. [cit. 2023-05-08]. Dostupné z: `https://ghidra-sre.org`

[12] POMAREDE, Nicolas et al. *Hatari* [online]. [cit. 2023-05-08]. Dostupné z: `http://hatari.tuxfamily.org`

[13] LENDINO, Jamie. *Faster Than Light: The Atari ST and the 16-Bit Revolution.* Audubon, NJ: Steel Gear Press, 2019, 325 s. ISBN 978-1732355217.

[14] MOTOROLA INC. *M68000 8-/16-/32-Bit: Microprocessors User's Manual* [online]. Ninth Ed. 1993, 189 s. [cit. 2023-05-08]. Dostupné z: https://www.nxp.com/docs/en/reference-manual/MC68000UM.pdf

[15] *MC68901 multifunction peripheral: technical summary.* In: MOTOROLA, INC. *MC68000 family reference manual*[online]. 9-25 - 9-71. [cit. 2023-05-08]. https://www.alldatasheet.com/datasheet-pdf/pdf/4169/MOTOROLA/MC68901.html

[16] ATARI (JAPAN) CORPORATION. *Atari 520ST Schematic* [online]. 1985, 3 s. [cit. 2023-05-08]. Dostupné z: https://archive.org/details/Atari_520ST_Schematic_1985/mode/2up

[17] HOMZOVÁ, Eliška. *Realizace výukového vícekanálového zvukového obvodu* [online]. Brno, 2021 [cit. 2022-12-15]. Dostupné z:http://hdl.handle.net/11012/198025. Bakalářská práce. Vysoké učení technické v Brně. Fakulta elektrotechniky a komunikačních technologií. Ústav automatizace a měřicí techniky. Vedoucí práce Petr Petyovský.

[18] GENERAL INSTRUMENT. *AY-3-8910, AY-3-8912, AY-3-8913: Programmable Sound Generator* [online]. 8 s. [cit. 2023-04-08]. Dostupné z: https://map.grauw.nl/resources/sound/generalinstrument_ay-3-8910.pdf. Datasheet.

[19] *Pulse Code Modulation*[online]. Tutorialspoint.com [cit. 2022-12-04]. Dostupné z:https://www.tutorialspoint.com/digital_communication/digital_communication_pulse_code_modulation.htm#

[20] *SILVEIRA, Marcelo. 8-bit sounds on MSX PSG* [online]. 2022, 11 s. [cit. 2022-12-20]. Dostupné z: http://marmsx.msxall.com/artigos/som_8bit_en.pdf

[21] RASPBERRY PI LTD. *Raspberry Pi Pico Datasheet: An RP2040-based microcontroller board* [online]. 2.1. 2023, 31 s. [cit. 2023-03-03]. Dostupné z:https://datasheets.raspberrypi.com/pico/pico-datasheet.pdf

[22] RASPBERRY PI LTD. *RP2040 Datasheet: A microcontroller by Raspberry Pi* [online]. 2.1. 2023, 639 s. [cit. 2023-04-07]. Dostupné z: https://datasheets.raspberrypi.com/pico/pico-datasheet.pdf

[23] *Power Electronics - Pulse Width Modulation* [online]. Tutorialspoint.com [cit. 2023-05-03]. Dostupné z:https://www.tutorialspoint.com/power_electronics/power_electronics_pulse_width_modulation.htm

[24] PONČÁK, Matej. *Inovace laboratorních úloh kurzu Vestavné systémy* [online]. Brno, 2022 [cit. 2023-03-10]. Dostupné z: http://hdl.handle.net/11012/204867. Diplomová práce. Vysoké učení technické v Brně. Fakulta elektrotechniky a komunikačních technologií. Ústav automatizace a měřicí techniky. Vedoucí práce Petr Petyovský.

Seznam symbolů a zkratek

ASCII	Americký standardní kód pro výměnu informací
GPIO	Víceúčelový vstupně/výstupní
I2C	Dvouvodičové komunikační rozhraní
MIDI	Digitální rozhraní pro hudební nástroje
PC	Osobní počítač
PCM	Pulzně kódová modulace
PWM	Pulzně šířková modulace
RTC	Hodiny reálného času
SPI	Sériové periferní rozhraní
SRAM	Statická paměť pro čtení i zápis
The Voder	The Voice operation demonstrator
TTS	Převod textu na řeč
UART	Univerzální asynchronní přijímač/vysílač
USART	Univerzální synchronní/asynchronní přijímač/vysílač
USB	Univerzální sériová sběrnice

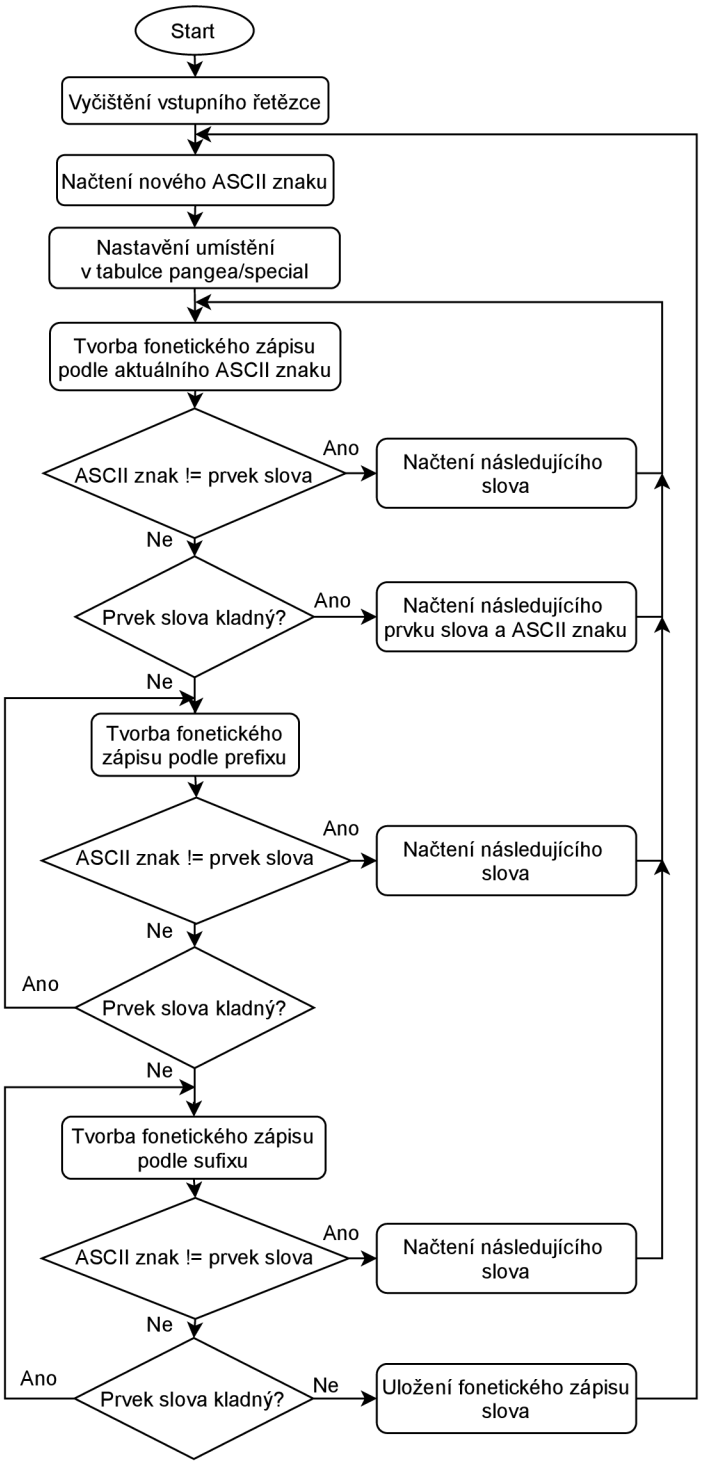
Seznam příloh

A Fonetické abecedy ARPAbet a IPA

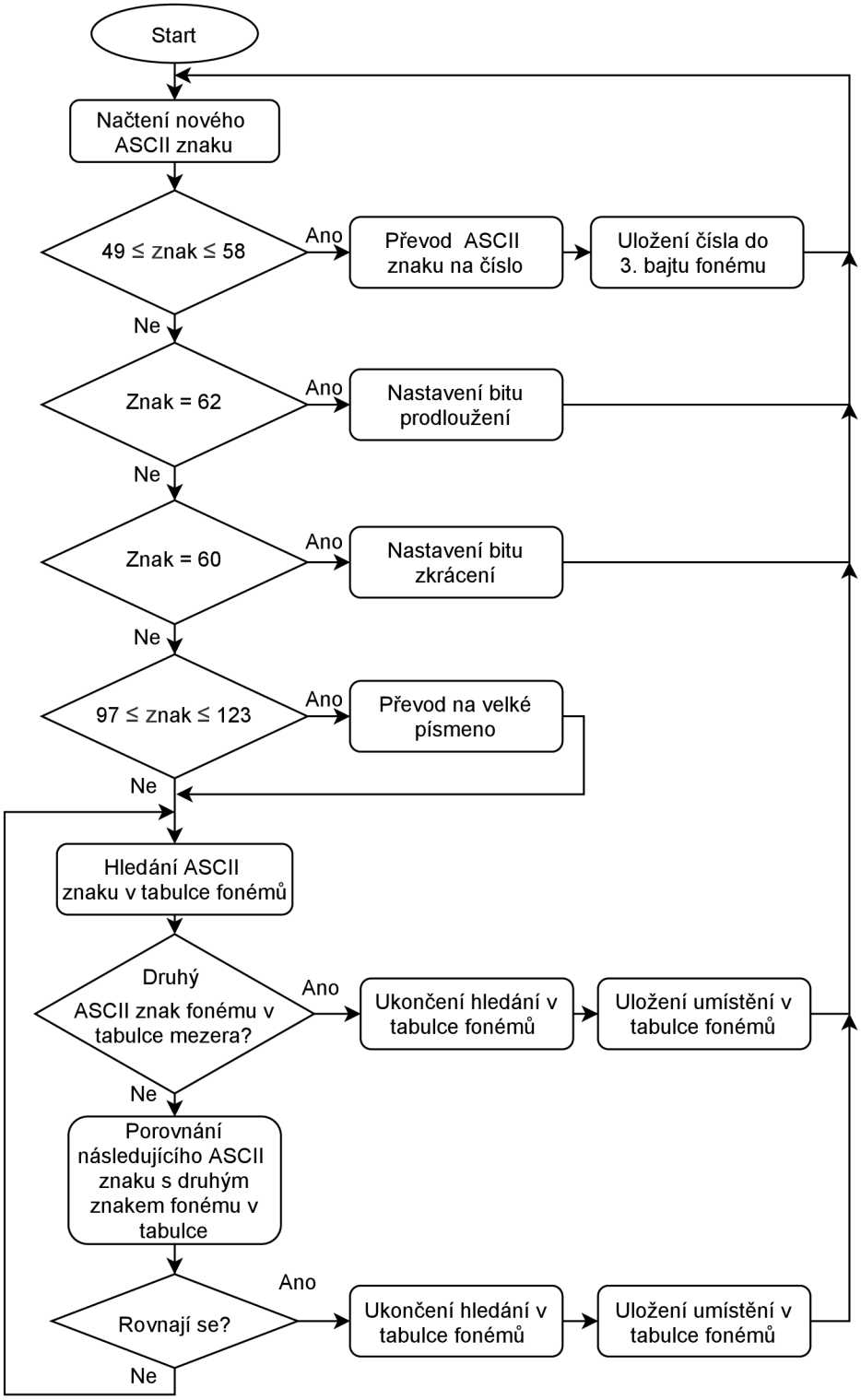
Tab. A.1: Symboly fonetických abeced ARPAbet a IPA pro anglické souhlásky a samohlásky[1]

IPA	ARPAbet	Příklad	IPA	ARPAbet	Příklad
p	p	parsley	w	w	kiwi
t	t	tea	r	r	rice
k	k	cook	j	y	yellow
b	b	bay	h	h	honey
d	d	dill	i	iy	lily
g	g	garlic	ɪ	ih	lily
m	m	mint	ey	ey	daisy
n	n	nutmeg	ɛ	eh	pen
ŋ	ng	baking	æ	ae	aster
f	f	flour	ɑ	aa	poppy
v	v	clove	ɔ	ao	orchid
θ	th	thick	ʊ	uh	wood
ð	dh	those	oʊ	ow	lotus
s	s	soup	u	uw	tulip
z	z	eggs	ʌ	ah	butter
ʃ	sh	squash	ɝ	er	bird
ʒ	zh	ambrosia	aɪ	ay	iris
tʃ	ch	cherry	aʊ	aw	flower
dʒ	jh	jar	oɪ	oy	soil
l	l	licorice			

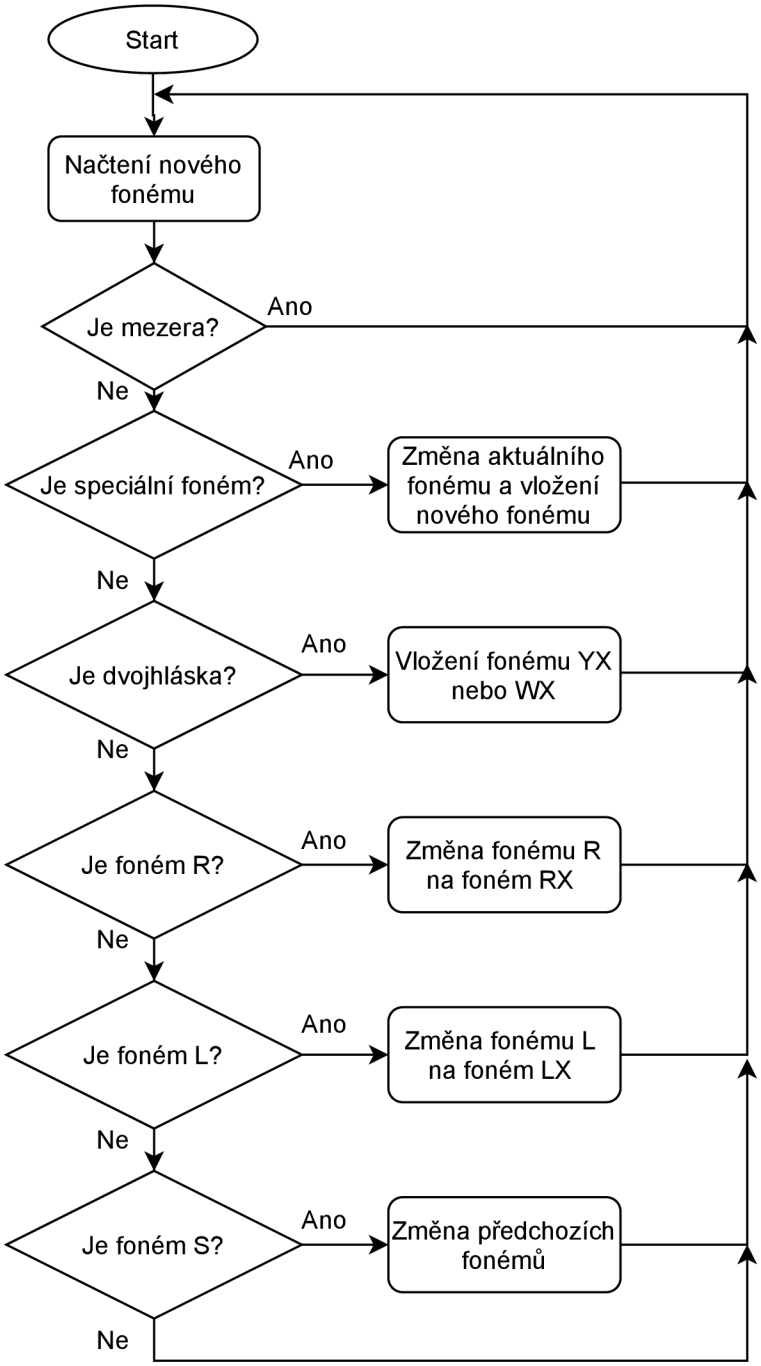
B Vývojové diagramy programu Atari 520ST Speech Synthesizer V2.0



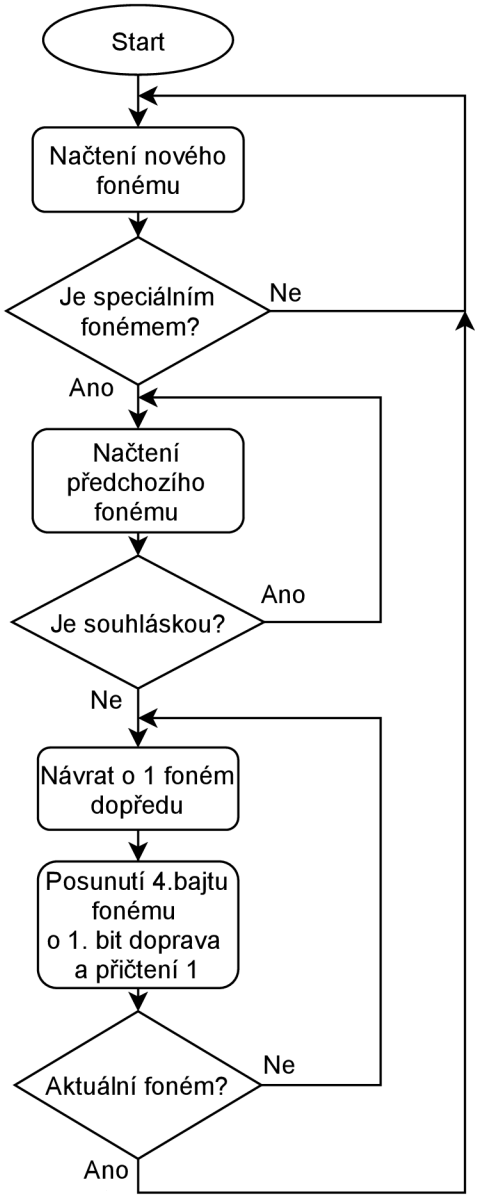
Obr. B.1: Vývojový diagram fonetické transkripce



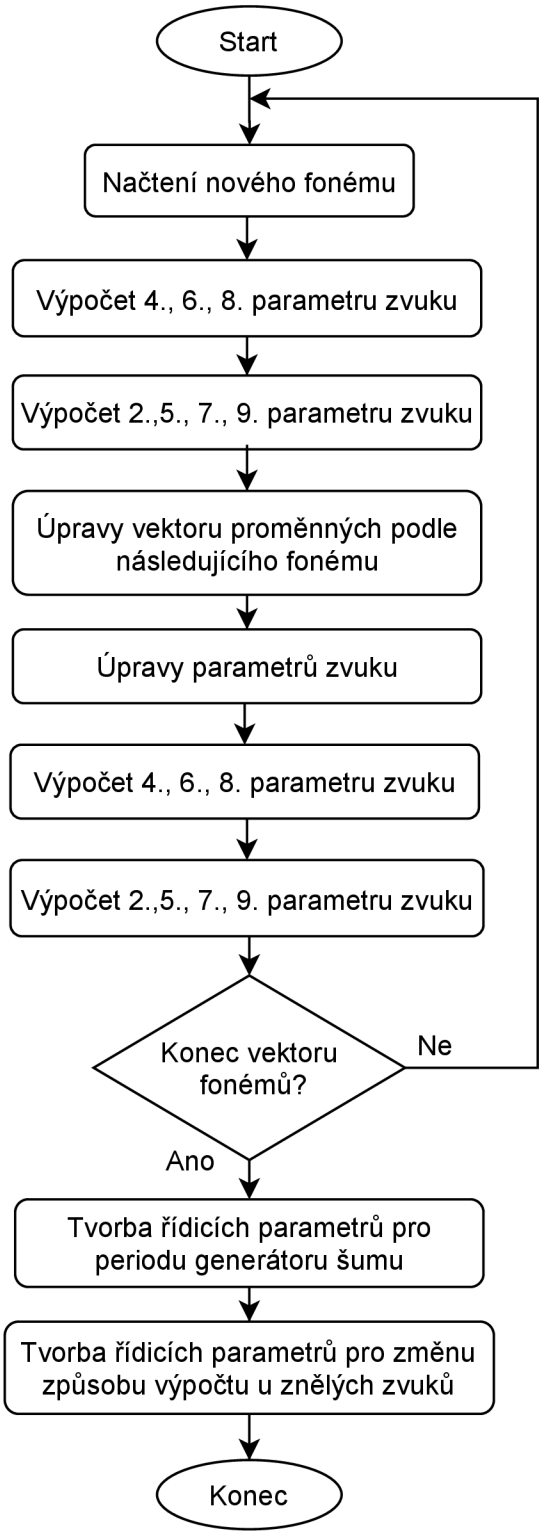
Obr. B.2: Vývojový diagram převodu fonetického zápisu na vektor fonémů



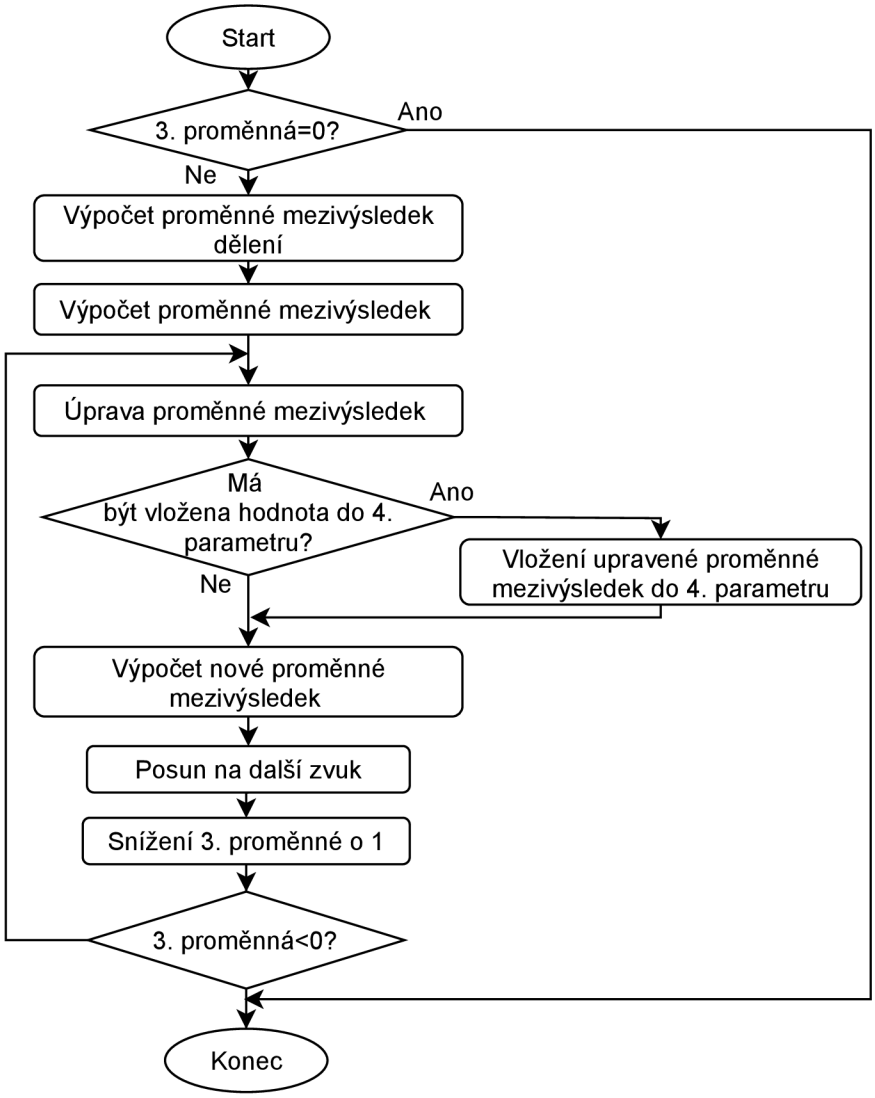
Obr. B.3: Vývojový diagram změny fonémů podle umístění ve slově



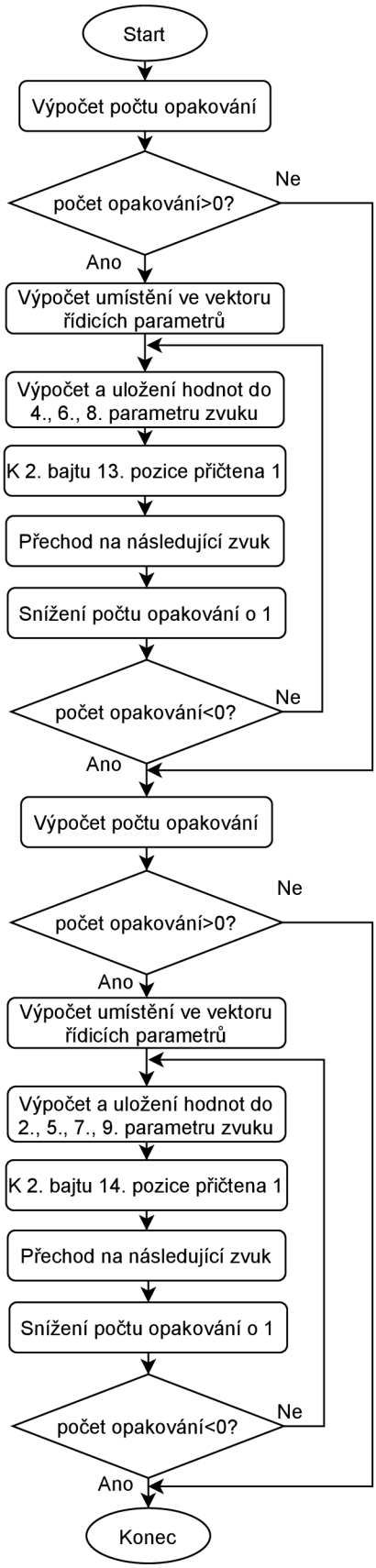
Obr. B.4: Vývojový diagram úpravy souhlásky před speciálním fonémem



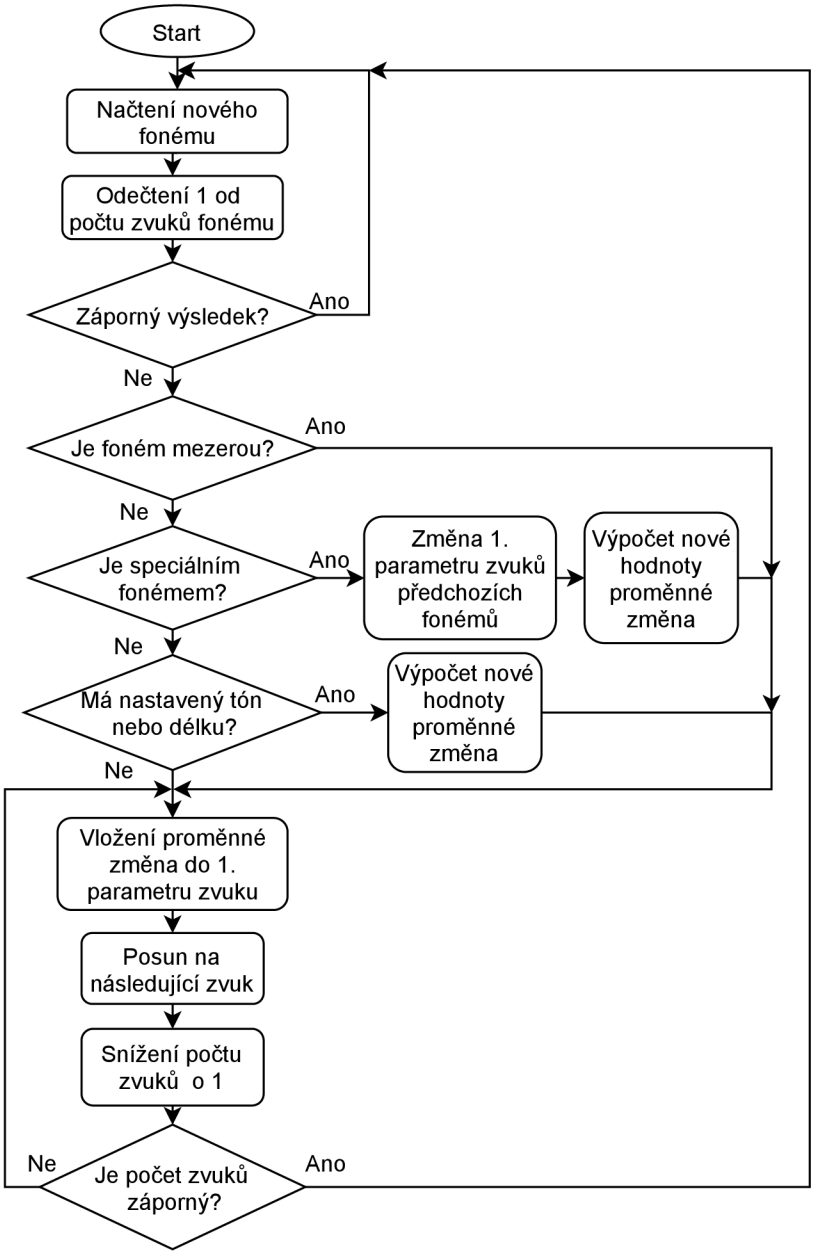
Obr. B.5: Vývojový diagram tvorby řídicích parametrů



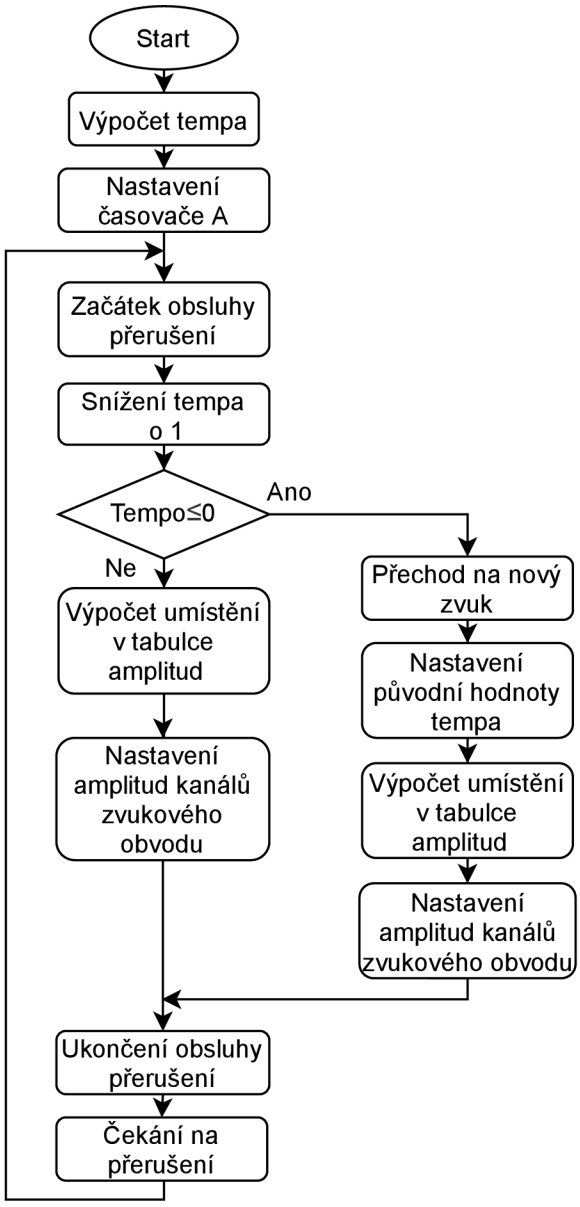
Obr. B.6: Vývojový diagram výpočtu 4. parametru zvuku vektoru řídicích parametrů



Obr. B.7: Vývojový diagram úpravy parametrů zvuku vektoru řídicích parametrů

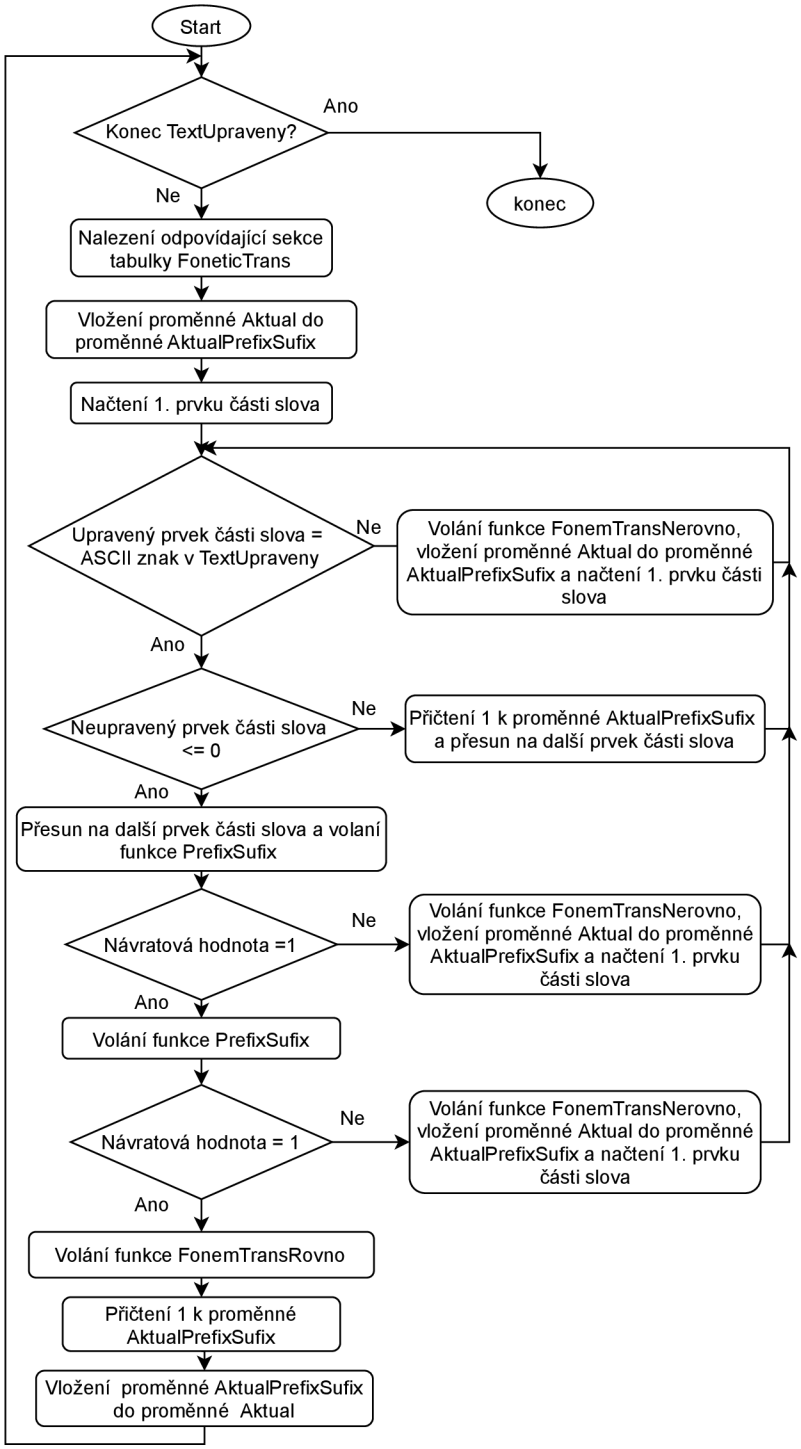


Obr. B.8: Vývojový diagram tvorby řídicích parametrů pro změnu způsobu výpočtu u znělých zvuků

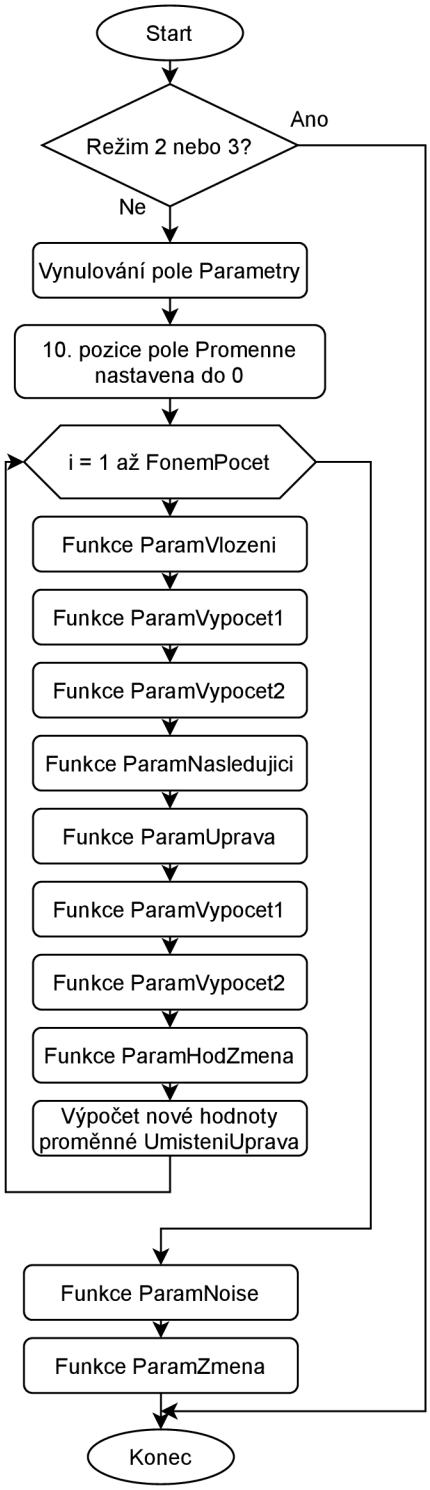


Obr. B.9: Vývojový diagram generování řeči

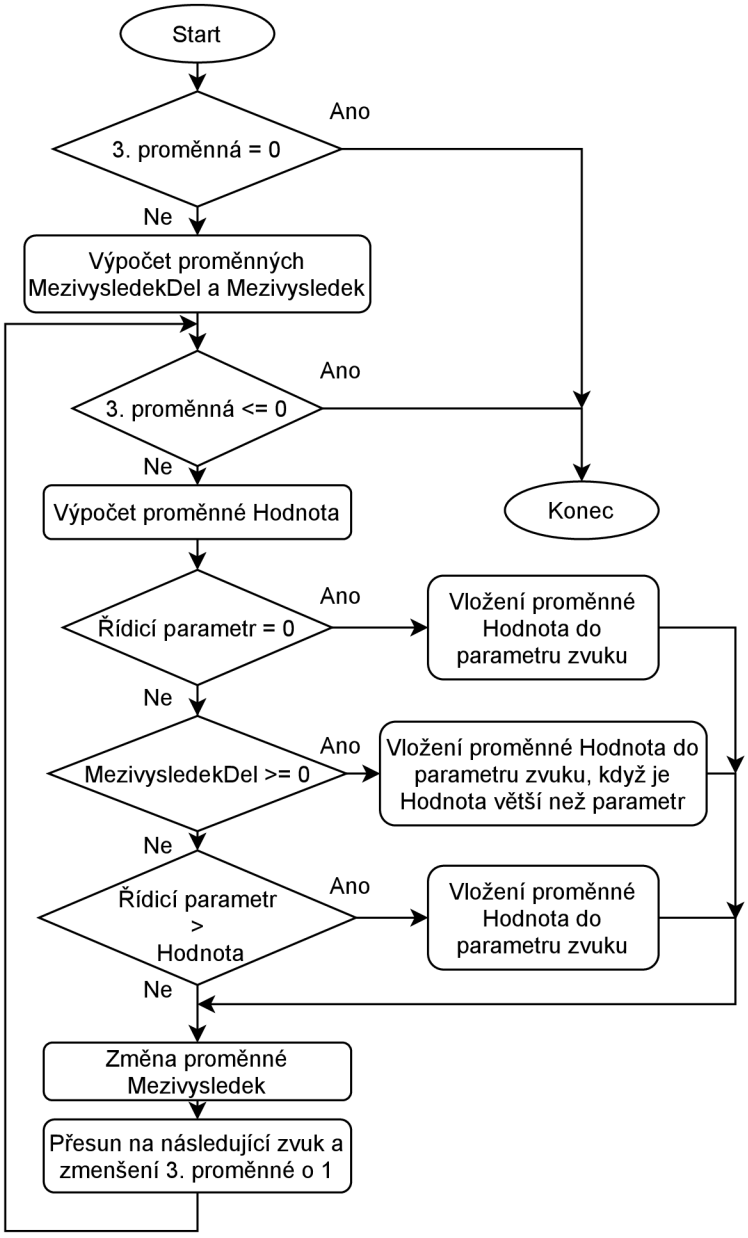
C Vývojové diagramy implementovaného hlasového generátoru



Obr. C.1: Vývojový diagram funkce FonemTransAlg

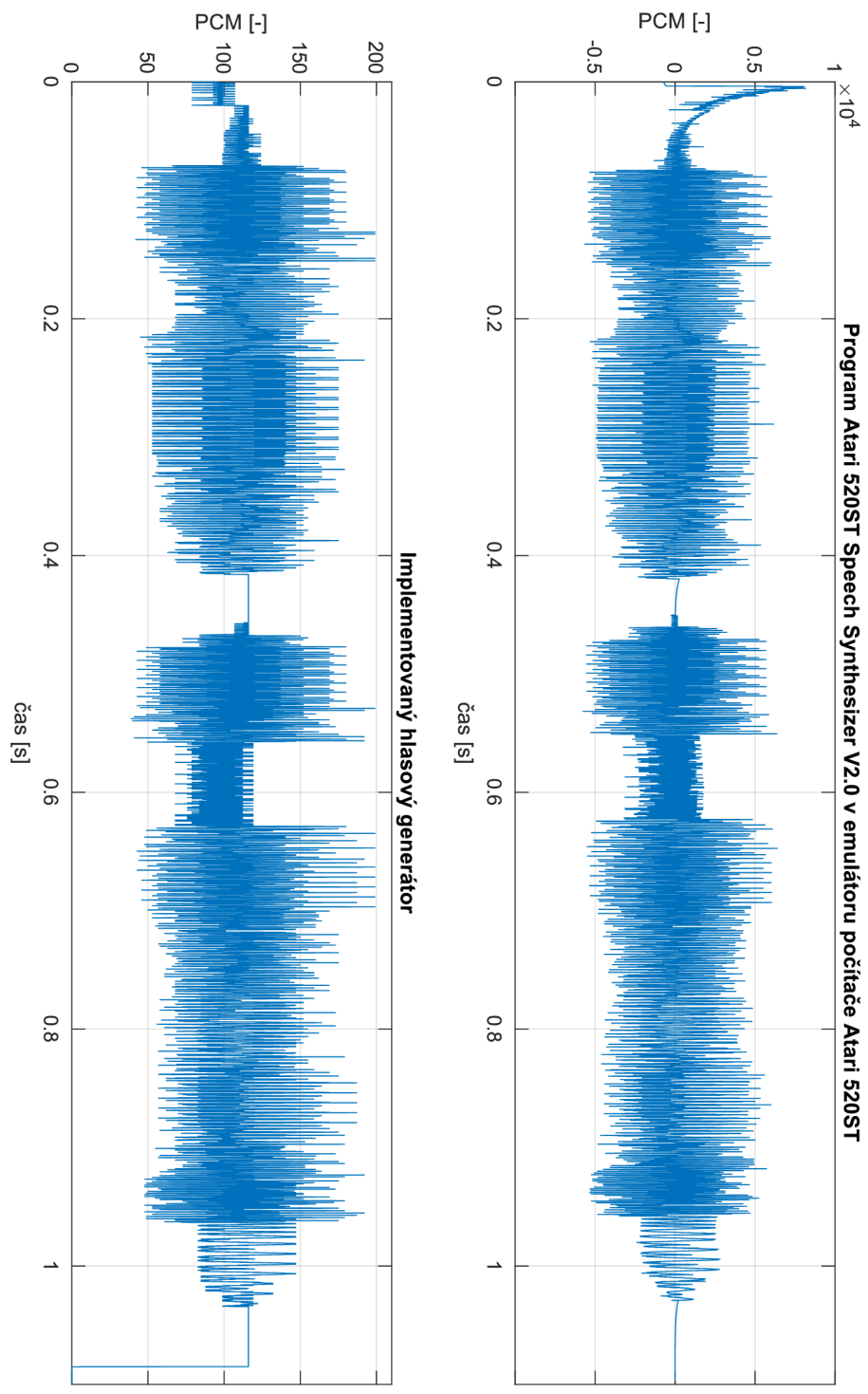


Obr. C.2: Vývojový diagram funkce RidiciParametry



Obr. C.3: Vývojový diagram funkce ParamVypocet1Alg

D Grafy průběhů generované řeči



Obr. D.1: Grafy průběhů slova "Hello" generovaného implementovaným hlasovým generátorem a programem v emulátoru Atari 520ST

E Obsah elektronické přílohy

- `/` ..... Kořenový adresář přiloženého archivu
  - `TTS` ..... Softwarová knihovna pro vybraný embedded systém
    - `TTS.h`
    - `TTS.cpp`
    - `Amplituda.cpp`
    - `AmplitudaUmisteni.cpp`
    - `FoneticTrans.cpp`
    - `ParametryFonemy.cpp`
  - `TTS_wav` ..... Demonstrační aplikace zápisu do souboru .wav
    - `TTS.h`
    - `TTS.cpp`
    - `Amplituda.cpp`
    - `AmplitudaUmisteni.cpp`
    - `FoneticTrans.cpp`
    - `ParametryFonemy.cpp`
    - `main.cpp`
  - `TTS_Raspberry` ..... Demonstrační aplikace na Raspberry Pi Pico
    - `TTS.h`
    - `TTS.cpp`
    - `Amplituda.cpp`
    - `AmplitudaUmisteni.cpp`
    - `FoneticTrans.cpp`
    - `ParametryFonemy.cpp`
    - `main.cpp`
    - `CMakeLists.txt`
    - `TTS_Raspberry.uf2`
  - `AtariProgram` ..... Přeložený strojový kód pouze na paměťovém médiu
  - `Nahravky` ..... Adresář s demonstračními nahrávkami
    - `HlasGen` ..... Nahrávky hlasového generátoru zápisem do souboru .wav
      - `HlasGen_85,77` ..... Tempo 85, Intonace 77
      - `HlasGen_100,65` ..... Tempo 100, Intonace 65
      - `HlasGen_100,77` ..... Tempo 100, Intonace 77
      - `HlasGen_100,100` ..... Tempo 100, Intonace 100
    - `Emulator` ..... Nahrávky původního programu z emulátoru Atari 520ST
      - `Emul_85,77` ..... Tempo řeči 85, Intonace 77
      - `Emul_100,65` ..... Tempo 100, Intonace 65
      - `Emul_100,77` ..... Tempo 100, Intonace 77
      - `Emul_100,100` ..... Tempo 100, Intonace 100
    - `Raspberry` ..... Nahrávky hlasového generátoru na Raspberry Pi Pico
      - `Raspberry_85,77` ..... Tempo 85, Intonace 77
      - `Raspberry_100,65` ..... Tempo 100, Intonace 65
      - `Raspberry_100,77` ..... Tempo 100, Intonace 77
      - `Raspberry_100,100` ..... Tempo 100, Intonace 100
  - `Thesis.pdf` ..... Elektronická verze diplomové práce